Departamento de Ingeniera Telemática

PROYECTO FIN DE CARRERA

DESARROLLO DE UNA APLICACIÓN DE PAGO A TRAVÉS DE LA TECNOLOGÍA NFC

Autor: Aida Campa Ruiz

Tutor: Mario Muñoz Organero

Leganés, Junio de 2011

Título: DESARROLLO DE UNA APLICACIÓN DE PAGO A TRAVÉS DE LA TECNOLOGÍA NFC
Autor: Aida Campa Ruiz
Director: Mario Muñoz Organero

EL TRIBUNAL

Presidente: Luis de la Fuente Valentín

Vocal: Ana Santos Rodríguez

Secretario: David Díez Hernández

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 3 de Junio de 2011 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO PRESIDENTE

Agradecimientos

En primer lugar, quiero expresar mi gratitud hacia Mario, el tutor de este proyecto, por su paciencia y por sus palabras de ánimo durante estos meses.

También a la empresa Gamma por la oportunidad que me brindaron de trabajar con ellos en esto, y a mi compañera de beca Bea, por los buenos momentos.

Hace ya algunos años, vine a Madrid para estudiar esta carrera. En Cantabria dejé a mi familia y amistades,

Quiero agradecer a los amigos que dejé allí, en especial a Cucho y a Digi, por siempre recibirme con alegría y demostrarme que, aún con el paso de los años, se me sigue echando de menos por allí.

Quiero agradecer a mi familia, enormemente, cómo me han animado y me han mandado fuerzas en cada una de las mañanas o tardes de examen, o en cada uno de los días de estudio. Gracias mamá, gracias papá, gracias David, gracias Pili, gracias Almu, gracias Paqui. A mis abuelos, a toda mi familia, a todos, gracias por apoyarme desde tan lejos. Os echo de menos siempre.

Gracias Ana e Iria, por estar siempre ahí, por todos los recuerdos que no nos quitan nadie, que tengo en la memoria y que me hacen sonreír.

A todos mis amigos, a todos, gracias. Hacéis que sienta que también tengo familia en Madrid. Sin poder extenderme mucho, quiero agradecérselo en especial a Pablete, por ayudarme y por hacerme tan feliz en tantos momentos. A Elena, por convertirse en este último año en una gran amiga. A Migue, por su apoyo y por sus palabras siempre reconfortantes cuando son necesarias. A Chemi, por ser mi amigo y cuidarme. A Adal, por estar siempre ahí y hacerme sentir que siempre estará ahí. A Jesús, por todos los momentos inolvidables. A Rober, por su forma de ser y por hacerme sentir tantas veces bien. A Deivid, también, por su incondicionalidad y apoyo. A Lisar, por la ayuda y simpatía. A Samu, por su amabilidad. A Pau, por ser como es, por ayudarme y darme tanto tantas veces. A Catherine, por su entrega y empatía. ¡Gracias!

Resumen

Los dispositivos móviles han adquirido en los últimos años una gran importancia en la sociedad de la comunicación y la información actual. Esta tendencia, se ha disparado con la aparición de los dispositivos *smartphones*.

La introducción a estos dispositivos de capacidades NFC, dota de un gran potencial a esta tecnología.

NFC una tecnología inalámbrica de corto alcance que permite comunicarse y obtener información entre dos dispositivos de forma muy fácil, con el simple gesto de acercar el uno con el otro.

A partir de esta característica, ya se le han encontrado multitud de usos para la sociedad, como, por ejemplo, recoger información de posters, tarjetas de fidelización de restaurantes, o realizar pagos.

En este proyecto se ha diseñado y desarrollado en Java, una simulación de un proceso de pago a través de la tecnología NFC integrada en un teléfono móvil. Al tratarse de una simulación de todo el proceso, se han implementado las distintas partes que intervienen en la compra y pago de un bien o servicio en una tienda, realizándose el pago a través de una entidad bancaria ficticia.

Palabras clave: NFC, Java, Pago móvil.

Abstract

Mobile devices have become in recent years of great importance in the current society of communication and information. This trend has exploded with the advent of smartphones devices.

The introduction of these devices with NFC capabilities provides this technology with great potential.

NFC is a wireless short-range technology that enables to communicate and to obtain information between two devices very easily, with the simple gesture of approaching each other.

Based on this feature, many uses for society have been found, for example, collecting information from posters, loyalty cards to restaurants, or making payments.

Throughout this project a simulation of a process of payment through NFC technology integrated in a mobile phone has been designed and developed in Java. Being this a simulation of the whole process, various parties involved in the purchase and payment for goods or services in a store have been implemented, making the payment through a fictitious bank.

Keywords: NFC, Java, Mobile Payment.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1 : Introducción y objetivos

1.1 Introducción

En este proyecto se ha desarrollado una simulación de un escenario de pago a través de la tecnología NFC.

A través de ello, se ha podido obtener una experiencia real con la que poder estudiar, y optimizar la experiencia de los distintos usuarios que intervienen en este prototipo:

- El usuario cliente, que posee un móvil con capacidad de comunicarse a través de NFC.

- El usuario vendedor, que posee un dispositivo o móvil lector con capacidad de comunicarse con el móvil cliente a través de NFC y con una red bancaria con la que gestionar el pago.

- El usuario gestor del banco, que a través de una interfaz web será capaz de gestionar las tarjetas del usuario cliente que utiliza para realizar pagos.

Figura 1: Ilustración de un pago con un móvil a través de la tecnología NFC, como ejemplo de su posible uso cotidiano

1.2 Objetivos

El objetivo principal de este proyecto trata de simular una operación de pago con un dispositivo móvil a través de la tecnología NFC.

En base a este objetivo principal, se ha desarrollado un protocolo de comunicación entre los distintos dispositivos que intervienen en el proceso, a través de las distintas formas de comunicación que poseen.

De una forma general, los objetivos relativos al objetivo principal del proyecto han sido:

- Análisis de la tecnología NFC, así como su estudio para su empleo en el proceso de comunicación entre dispositivos móviles.

- Estudio de la tecnología Bluetooth para su empleo en la comunicación entre distintos dispositivos.

- Desarrollo de aplicaciones en J2ME y J2SE.

- Estudio e implementación de un sistema de base de datos con una interfaz web, en el que poder albergar y modificar datos que intervengan en el proceso de comunicación entre los dispositivos móviles.

1.3 Fases del desarrollo

El desarrollo de este proyecto se ha realizado atravesando, no siempre de forma cronológica, distintas fases. Cada fase cubre distintos aspectos del desarrollo.

- Fase 0: Investigación y documentación sobre NFC y Bluetooth.
- Fase 1: Despliegue y configuración del entorno de programación y del sistema.
- Fase 2: Estudio y diseño de la funcionalidad y de los protocolos de las aplicaciones a realizar.
- Fase 3: Desarrollo de la funcionalidad en la que interviene la tecnología NFC.
- Fase 4: Desarrollo de la funcionalidad en la que interviene la tecnología Bluetooth.
- Fase 5: Integración de la funcionalidad NFC y Bluetooth.
- Fase 6: Diseño y desarrollo de la base de datos del sistema diseñado.
- Fase 7: Desarrollo aplicación web de acceso a la base de datos.
- Fase 8: Integración del sistema completo.
- Fase 9: Realización de pruebas y depuración del sistema completo.
- Fase 10: Documentación y redacción de la memoria.
- Fase 11: Corrección de la memoria.

1.4 Estructura de la memoria

Para facilitar la lectura de la memoria, se incluye a continuación un breve resumen de cada capítulo, para entender su estructura:

En el segundo capítulo, se recoge el Estado del Arte del proyecto, en el que se profundiza en los aspectos teóricos de las distintas tecnologías que se han utilizado para la consecución de este proyecto: NFC y Bluetooth. A continuación, se analiza el lenguaje Java, J2ME y Java Swing. También se hace una breve introducción de las tecnologías del lado del servidor. Por último, se realiza una descripción el teléfono móvil utilizado, Nokia 6131 NFC.

En el tercer capítulo, se realiza una descripción técnica sobre las implementaciones realizadas utilizando las tecnologías NFC y Bluetooth.

En el cuarto capítulo, se presenta una descripción funcional del sistema completo del proyecto.

En el quinto capítulo, se describe en detalle la implementación del sistema, y se recoge un breve manual de usuario

En el sexto capítulo, se recogen las pruebas que se han realizado para optimizar y validar el correcto funcionamiento de todas funcionalidades desarrolladas en el proyecto.

En el séptimo capítulo, se recoge las conclusiones a las que se ha llegado con la finalización del proyecto, así como las múltiples líneas de futuro que aún se pueden desarrollar.

En el octavo capítulo, se presenta el presupuesto total del proyecto.

Por último, el Anexo A, recoge la guía de instalación del Software para poder instalar y desplegar todo el escenario con el que se ha desarrollado este proyecto.

Capítulo 2 : Estado del Arte

En el presente capítulo se realiza la descripción y análisis de las distintas tecnologías utilizadas, como son NFC o Bluetooth.

También se detallan aspectos del lenguaje de programación utilizado, Java, y de la plataforma que se usa para el desarrollo de las aplicaciones móviles, J2ME.

Otro aspecto que se detalla, son las tecnologías y servicios utilizados para la realización de la base de datos y la interfaz web: *JSPs, Servlets* (Java), *JavaDB Apache Derby, Apache Tomcat.*

Por último se realizará una presentación al teléfono móvil Nokia 6131 NFC, utilizado para la realización de este proyecto.

2.1 La tecnología NFC

2.1.1 Introducción

El teléfono móvil se ha convertido en una herramienta diaria e indispensable en el que convergen muchas aplicaciones y tecnologías.

La tecnología inalámbrica NFC aparece como progreso en la convergencia de aplicaciones dentro del teléfono móvil. La provisión de servicios, el rango de aplicaciones, la confidencialidad, la velocidad, el coste, o la robustez son las características más importantes de esta tecnología frente al resto de propuestas inalámbricas.

El interés que NFC está generando tiene que ver con el potencial que ofrece para el desarrollo e implementación de novedosas e interesantes aplicaciones, sobre todo de cara a mejorar la experiencia del usuario en la utilización de servicios ya existentes, como es el caso del pago a través del móvil, que es precisamente del objeto de estudio de este proyecto.

Figura 2: Logo de la tecnología NFC

2.1.2 Historia

Para poder describir la tecnología NFC, es importante señalar primero que se trata de una técnica basada en *RFID* (*Radio Frequency Identification*). Dentro de este apartado se hace una breve introducción a la tecnología RFID. A continuación, se muestra una breve comparativa entre ambas tecnologías para entender sus diferencias.

Por último, se habla del NFC Fórum, creado el año 2004 por las tres compañías pioneras de la tecnología NFC: Philips, Sony y Nokia, para avanzar en el uso de la tecnología a través del desarrollo de especificaciones.

2.1.2.1 RFID

RFID (Identificación por Radiofrecuencia), es un método automático de identificación que se basa en el almacenamiento y captura remota de datos usando dispositivos llamados *tags (*etiquetas*)* RFID.

Una etiqueta RFID es un dispositivo pequeño, similar a una pegatina, que puede ser adherida o incorporada a un producto, animal o persona, con el propósito de identificarlo a distancia usando ondas de radio.

La implementación de sistemas RFID ha empezado a desarrollarse y a conocerse debido a su reciente masificación y abaratamiento de costes.

Actualmente, la aplicación más importante de RFID es la logística. El uso de esta tecnología permitiría tener localizado cualquier producto dentro de la cadena de suministro.

Figura 3: Ejemplo de aplicación de la tecnología RFID

Trabaja en diferentes bandas de frecuencias que van desde bandas de baja frecuencia (KHz) hasta bandas de alta frecuencia (GHz).

Los sistemas RFID se componen básicamente de tres elementos: Etiqueta RFID, Reader RFID y un Sistema de procesamiento de datos.

2.1.2.1.1 Etiqueta RFID

Compuesta por una antena, un transductor de radio y un material encapsulado o chip. Los hay de diferentes tipos, se pueden clasificar en pasivas y activas, y en solo lectura, o lectura y escritura.

Para las etiquetas activas, su fuente de alimentación es propia mediante baterías de larga duración. La duración de estas depende del modelo y de la actividad que tenga, pero suele ser de varios años. Además, generalmente, envían la información del estado de las baterías para que pueda haber un control de éstas.

Las etiquetas pasivas no poseen alimentación eléctrica. La señal que les llega de los lectores les induce una corriente eléctrica pequeña y suficiente para operar el circuito integrado CMOS la etiqueta, de forma que puede generar y transmitir una respuesta.

Las etiquetas cuya funcionalidad es “sólo lectura” (*Read Only*) o “escritura única y escritura múltiple” (*WORM, Write Once-Read Many*) son numeradas previamente y necesitan un banco de datos que reciba sus informaciones. Una vez programada, la etiqueta no puede ser modificada.

Las etiquetas de lectura-escritura (*Read-Write*) tienen una capacidad más grande y se pueden modificar o actualizar siempre que se necesite.

La memoria interna generalmente es de 4 y 32 *kbytes*.

Figura 4: Ilustración de las múltiples formas de presentación de las etiquetas RFID

2.1.2.1.2 Reader RFID

Compuesto de antenas, un transceptor y un decodificador. El Reader envía periódicamente señales para ver si hay alguna etiqueta en sus inmediaciones. Cuando capta una señal de una etiqueta, extrae dicha información y se la pasa al sistema de procesamiento de datos.

Tiene distintas distancias para la lectura y escritura de sus etiquetas, y pueden llegar generalmente hasta los 100m.

2.1.2.1.3 Sistema de procesamiento de datos

Es por lo general una aplicación que gestiona y procesa los datos recibidos del Reader RFID.

2.1.2.2 NFC Y RFID

Uno de los factores importantes que le falta a RFID y le diferencia de NFC, es la comunicación o transferencia de información entre dos dispositivos activos.

NFC logra justamente esto que no consigue RFID, y difiere de éste principalmente por dos motivos:

- NFC provee comunicación P2P (*peer-to-peer*), lo que le permite a dos dispositivos interconectarse, haciéndola mucho más potente.

- NFC no puede ser activado remotamente por accidente o involuntariamente. El teléfono obliga a que deba existir un acercamiento entre dispositivos antes de iniciar una comunicación.

NFC combina la característica de leer y emular etiquetas RFID, y también de intercambiar datos entre dispositivos electrónicos que tengan carga de energía.

2.1.2.3 NFC FORUM

Aunque la tecnología NFC está bien posicionada para su adopción por una gran multitud de fabricantes de la electrónica de consumo, el *Near Field Communication Forum* fue formado para avanzar en el uso de la tecnología a través del desarrollo de especificaciones, para poder así asegurar la interoperabilidad entre dispositivos y servicios y educar al mercado sobre esta tecnología. *[1], [2], [3]*

Figura 5: Logo del NFC Forum

Fue fundado en el año 2004 por las tres compañías pioneras: Philips, Sony y Nokia, y actualmente cuenta con 140 miembros.

Fabricantes, desarrolladores de aplicaciones, instituciones financieras, vendedores, agencias gubernamentales, consorcios de transporte y

organizaciones sin ánimo de lucro, están trabajando conjuntamente para promover el uso de la tecnología NFC en electrónica de consumo, dispositivos móviles y ordenadores.

Los objetivos del NFC Forum son:

- Crear especificaciones basadas en estándares para los desarrolladores, que definan una arquitectura modular y unos parámetros de interoperabilidad para todos los dispositivos y protocolos NFC.

- Fomentar el desarrollo de productos usando las especificaciones del NFC Forum.

- Trabajar para asegurar que los productos que requieren capacidades NFC cumplan con las especificaciones del NFC Forum.

- Enseñar y educar a las empresas y a los consumidores sobre esta tecnología.

El NFC Forum proporciona un marco de trabajo estable para el desarrollo de aplicaciones seguras e interoperables entre sí. Para conseguir este objetivo, se desarrollan especificaciones que definen la arquitectura de los dispositivos NFC y los protocolos para la interoperabilidad. Urge a sus miembros el uso de las especificaciones desarrolladas, y trabaja para que todos los dispositivos bajo la denominación NFC cumplan con las especificaciones del Forum NFC garantizando así un correcto funcionamiento.

En Junio del 2006, sólo 18 meses después de su fundación, el Forum formalmente perfiló la arquitectura de la tecnología NFC.

En Septiembre del 2010, el Forum había publicado ya 13 especificaciones y 3 candidatas a especificación. Las especificaciones proporcionan un esquema temporal o "*roadmap*", que permite que a los interesados puedan crear y desarrollar productos interesantes para sus clientes.

Actualmente la colaboración técnica y de negocios relacionada con NFC se realiza bajo los acuerdos del NFC Forum. Sus miembros principales son: NXP (Philips), Sony, Nokia, HP, Texas Instruments, NEC, Samsung, Motorola, MasterCard, Visa, Panasonic, Microsoft, Gemalto, Vodafone, NTT DoCoMo, Siemens, entre otros.

2.1.3 Descripción general NFC

NFC es una tecnología inalámbrica de corto alcance basada en la tecnología RFID, que permite realizar una comunicación simple, segura e intuitiva

entre dispositivos electrónicos que se encuentran a una distancia de hasta de 10 centímetros en algunos casos. *[4]*

Dentro de la forma de trabajo de NFC, un dispositivo genera una onda de radio de baja frecuencia que opera a 13,56 MHz. Cuando otro dispositivo NFC se acerca lo suficiente para ponerse en contacto, se genera un “acoplamiento magnético inductivo”, por medio del cual se puede realizar una transferencia de energía y de datos entre los dispositivos.

Figura 6: Ejemplo de uso de un teléfono NFC para realizar un pago, a través de un lector NFC

Este acoplamiento inductivo funciona sólo para distancias cortas y es por esto que el uso de este tipo de acoplamiento es la principal diferencia entre NFC y otras tecnologías inalámbricas como Bluetooth y WiFi, donde éstas trabajan a una distancia máxima aproximada de 10 metros y 100 metros, respectivamente.

Entre los beneficios más importantes que se pueden señalar del uso y desarrollo de esta tecnología, se destacan los siguientes:

- Mejora la usabilidad y la experiencia del usuario.
- Fácil acceso a servicios y contenidos ofrecidos por objetos físicos.
- Se puede compartir información digital entre dos dispositivos con tan sólo acercar el uno al otro.
- Seguridad, por ser una tecnología de corto alcance.

2.1.3.1 Definición estándar de datos NFC

Para poder compartir datos entre los dispositivos NFC entre sí y/o entre los dispositivos y las etiquetas NFC, se ha creado un estándar en el que se registra un formato común.

- NFC Data Exchange Format (NDEF): Especifica un formato común y compacto para el intercambio de datos.

- NFC Record Type Definition (RTD): Especifica tipos de registros estándar que pueden ser enviados en los mensajes intercambiados entre los dispositivos NFC.

    - Smart Poster RTD: Para posters que incorporen etiquetas con datos (URLs, SMSs o números de teléfono).

    - Text RTD: Para registros que solo contienen texto.

    - Uniform Resource Identifier (URI) RTD: Para registros que se refieren a un recurso de Internet.

2.1.3.2 Especificaciones técnicas

Opera dentro de la banda ISM (*Industrial, Scientific and Medical*) de radio frecuencia de 13,56 MHz disponible globalmente sin restricción y sin necesidad de licencia para su uso, con un ancho de banda de casi 2 MHz.

NFC es una tecnología de plataforma abierta estandarizada en la ISO/IEC 18092 y la ECMA-340. Estos estándares especifican los esquemas de modulación, codificación, velocidades de transferencia y formato de la trama de la interfaz RF de dispositivos NFC, así como los esquemas de inicialización y condiciones requeridas para el control de colisión de datos durante la inicialización para ambos modos de comunicación, activo y pasivo. También definen el protocolo de transporte, incluyendo los métodos de activación de protocolo y de intercambio de datos.

La distancia de trabajo con antenas compactas estándar es aproximadamente 20 cm, aunque generalmente es efectivo cercano a los 10 cm. Las velocidades de transmisión que soporta esta tecnología son de 106, 212, 424 u 848 kbits/s.

La comunicación NFC es bidireccional, por lo tanto los dispositivos NFC son capaces de transmitir y recibir datos al mismo tiempo.

2.1.3.3 Modos de funcionamiento

Un dispositivo NFC con suministro interno de energía es denominado activo, y sin suministro interno de energía, como una etiqueta, es considerado pasivo. El acoplamiento inducido causa que un dispositivo pasivo absorba energía de uno activo cuando se acercan lo suficiente. Una vez encendido, el dispositivo pasivo puede comunicarse e intercambiar datos con el otro dispositivo.*[5]*

La habilidad de funcionar en los dos modos, activo o pasivo, hace que los dispositivos NFC sean únicos dentro de otras tecnologías de comunicación sin contacto. Esto posibilita a los dispositivos a actuar como tarjetas sin contacto o como lectores. Por tanto, un teléfono móvil habilitado con NFC puede ser usado por ejemplo, para enviar información de pago a un lector y realizar una compra o para leer información de una valla o poster publicitario con una etiqueta adherida.

La descripción de ambos modos de funcionamiento se detalla a continuación:

2.1.3.3.1 Pasivo

Sólo un dispositivo genera el campo electromagnético y el otro se aprovecha de la modulación de la carga para poder transferir los datos. El iniciador de la comunicación es el encargado de generar el campo electromagnético.

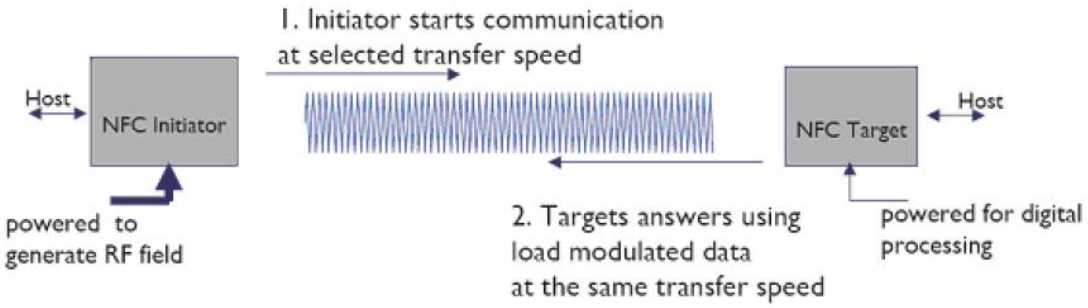

Figura 7: Proceso de comunicación en modo pasivo

2.1.3.3.2 Activo

Ambos dispositivos generan su propio campo electromagnético, que utilizarán para transmitir sus datos. Ambos dispositivos necesitan energía para funcionar.

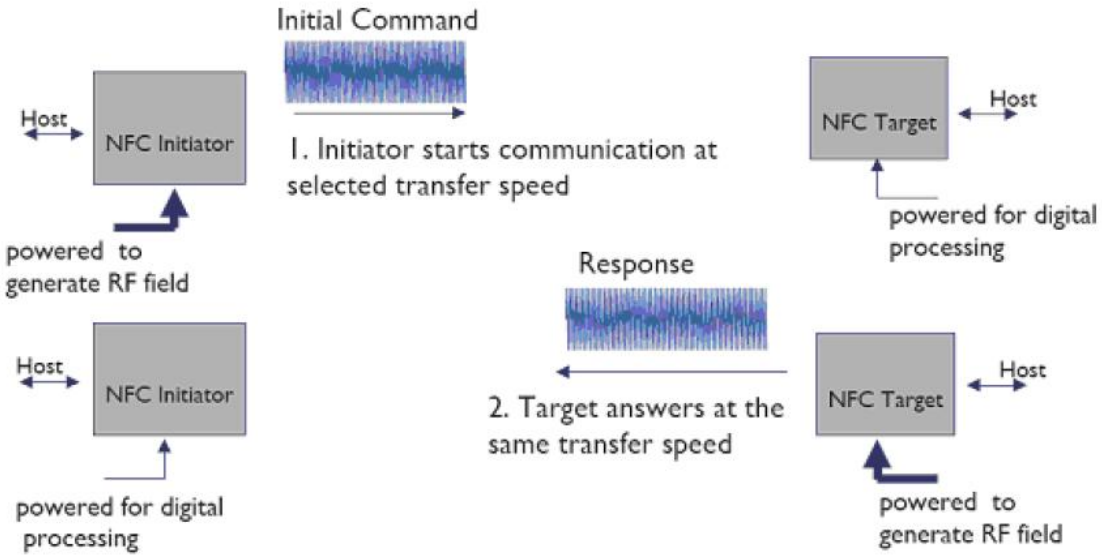

Figura 8: Proceso de comunicación en modo activo

Cualquier dispositivo electrónico con NFC (excepto una etiqueta NFC) puede operar de las dos formas.

2.1.3.4 Transacción NFC

La comunicación NFC consta de cinco fases, estando siempre presentes. Estas etapas son:

- **Descubrimiento**: En esta fase los dispositivos inician la etapa de rastrearse el uno al otro y posteriormente su reconocimiento.

- **Autenticación:** En esta parte los dispositivos verifican si el otro dispositivo está autorizado o si deben establecer algún tipo de cifrado para la comunicación.

- **Negociación**: En esta parte del establecimiento, los dispositivos definen parámetros como la velocidad de transmisión, la identificación del dispositivo, el tipo de aplicación, su tamaño, y si es el caso también definen la acción a ser solicitada.

- **Transferencia**: Una vez negociados los parámetros para la comunicación, se puede decir que ya está realizada exitosamente la comunicación y ya se puede realizar el intercambio de datos.

- **Confirmación**: El dispositivo receptor confirma el establecimiento de la comunicación y la transferencia de datos.

La tecnología NFC no está destinada para la transferencia masiva de datos, pero se puede utilizar para la configuración de otras tecnologías inalámbricas de mayor ancho de banda como Bluetooth *o Wi-Fi* con la ventaja de que si se utiliza NFC el tiempo de establecimiento de la comunicación es muy inferior que si se utilizaran estas otras tecnologías por sí solas para efectuar el enlace.

2.1.3.5 Arquitectura y modalidades de utilización

Podemos decir que NFC es una tecnología única ya que puede trabajar en tres diferentes configuraciones lo que la hace que sea más adaptable y eficiente que otras.

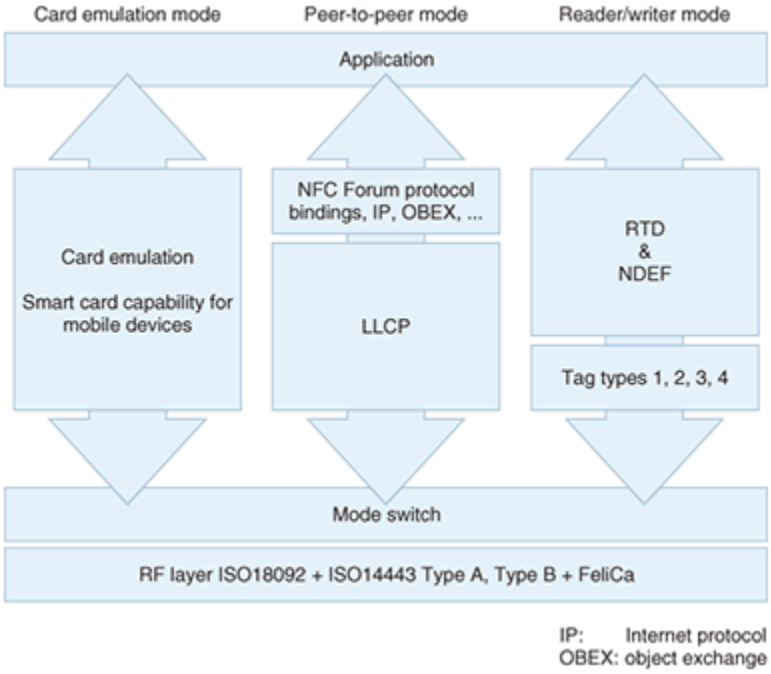

Figura 9: Las tres diferentes configuraciones en las que NFC puede trabajar

La tecnología NFC puede trabajar en tres distintas configuraciones, tal y como se muestra en la Figura 9.

Una visión simplificada de estas tres configuraciones la tenemos en la Figura 10:

- En el modo Lector/(Escritor) un dispositivo NFC lee información de una etiqueta NFC.

- En el modo Emulación de Tarjeta el dispositivo NFC se comporta como una etiqueta NFC y es leído por otro dispositivo.
- En el modo Peer-to-peer dos dispositivos se intercambian información de igual a igual.

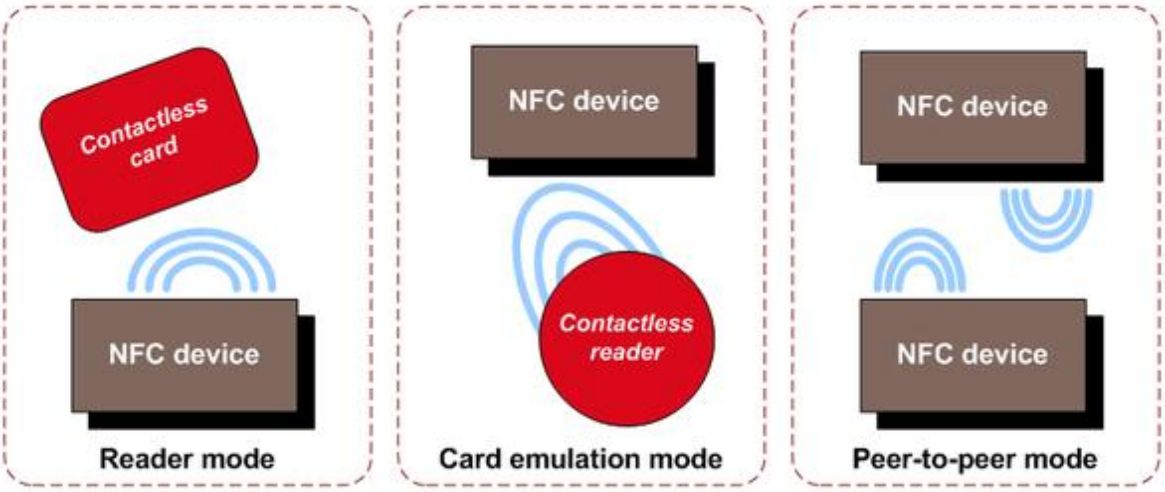

Figura 10: Visión simplificada de las tres configuraciones en las que NFC puede trabajar

A continuación, se describen estas tres configuraciones con más detalle:

2.1.3.5.1 Modo Emulación de Tarjeta Inteligente NFC

Este modo se utiliza para que el dispositivo NFC actúe como una etiqueta o una tarjeta inteligente, apareciendo ante un lector externo como si se tratase de una tarjeta sin contacto.

Con esta configuración también se puede utilizar las características de seguridad avanzadas del elemento seguro, siendo útil para, por ejemplo, transacciones bancarias, o gestión de entradas y accesos.

En la Figura 11, se muestra la torre de protocolos de este modo de funcionamiento, en el que se releva la gestión de la lectura de datos para el nivel de aplicación.

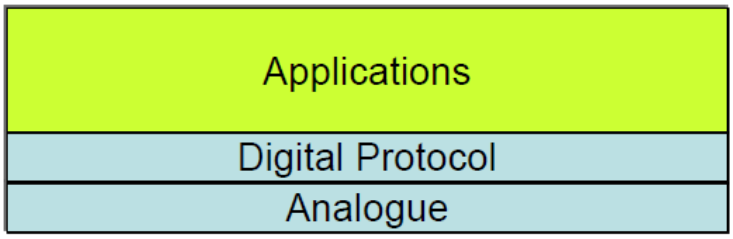

Figura 11: Torre de protocolos para el modo emulación de tarjeta inteligente NFC

2.1.3.5.2 Modo de Comunicación Peer-to-Peer

Este modo sirve para el intercambio de pequeñas cantidades de datos utilizando el mismo protocolo de NFC.

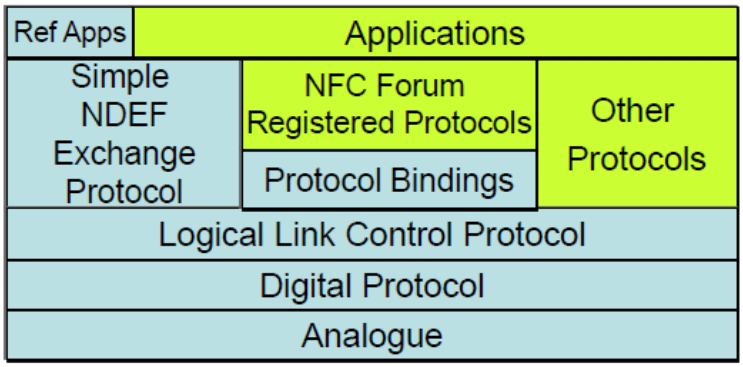

Figura 12: Torre de protocolos para el modo de comunicación Peer-to-peer

En la Figura 12 se muestra la torre de protocolos de este modo de funcionamiento: NFC utiliza a nivel de la capa de enlace el protocolo de control de enlace lógico (LLCP), el mismo que es usado para la activación, supervisión y desactivación de la comunicación.

El modo de transferencia se hace de modo asincrónico balanceado, es decir que cualquier dispositivo puede iniciar la transmisión sin permiso de la otra.

El protocolo de intercambio simple NDEF se utiliza para enviar mensajes con el formato NDEF en este modo.

Los protocolos de conexión "*Bindings*" proporcionan enlaces estándar para protocolos NFC registrados y permite su uso interoperable.

Los protocolos NFC registrados son aquellos para los que el *NFC Forum* ha definido un enlace con la capa LLCP (Protocolo de Control de Enlace Lógico) , por ejemplo IP, OBEX (intercambio de Objetos binarios), etc.

Las aplicaciones en modo *Peer-to-Peer* podrían ser, por ejemplo, imprimir desde una cámara, intercambiar una tarjeta de negocios, intercambiar imágenes entre dos móviles, etc.

2.1.3.5.3 Modo Lectura / Escritura

En este modo, el dispositivo NFC es capaz de escribir y leer los cuatro tipos de etiquetas especificados por el NFC Forum, que se detallan en el siguiente apartado.

En la torre de protocolos que podemos observar en la Figura 13, se muestra que se puede trabajar con cualquier tipo de etiqueta, y utilizando, o no, el modo de definición de datos NDEF.

En esta configuración, cuando el usuario toca con su dispositivo con tecnología NFC una tarjeta o etiqueta NFC, se transfiere una pequeña cantidad de información al dispositivo NFC. Esta información puede ser un texto en claro, una dirección de una página web o un número de teléfono.

Un ejemplo de uso de esta configuración es el denominado póster inteligente, donde el usuario toca con su móvil NFC la etiqueta incorporada en el póster. Esta acción transmite al teléfono una dirección de una página web abriendo automáticamente el navegador web del teléfono con la página solicitada.

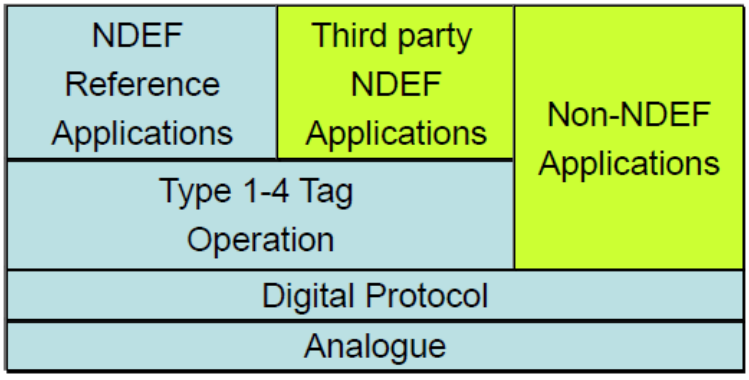

Figura 13: Torre de protocolos para el modo lectura/escritura

2.1.3.6 Etiquetas (tags) NFC

El NFC Forum ha creado un juego de cuatro formatos de etiquetas que todos los dispositivos acordes con el NFC Forum. Estos formatos están basados en el ISO 14443 Tipo A y B (Estándares internacionales para tarjetas inteligentes sin contacto) y FeliCa (derivado el ISO 18092, modo de comunicación pasivo). Algunas etiquetas compatibles con estos estándares están disponibles inicialmente por Innovision, Philips, Sony y otros fabricantes. En todo el mundo ya hay alrededor de un billón de etiquetas ya en uso.

Las etiquetas utilizadas en la tecnología NFC tienen un número de identificación único (con un tamaño de 4 a 10 bytes), que se da por la combinación entre código de empresa fabricante y tipo de tecnología de la etiqueta (MIFARE, FeliCa, etc.).

- Etiqueta Tipo 1: Se basa en la norma ISO14443A. Las etiquetas permiten ser de lectura y reescritura, aunque los usuarios pueden configurarlas sólo para lectura. La memoria disponible es de *96* Bytes y expandible a 2 Kb; la velocidad de comunicación es de 106 Kbps.

- Etiqueta Tipo 2: Se basa en la norma ISO14443A. Permiten ser de lectura y reescritura, aunque los usuarios pueden configurarla sólo para lectura. La memoria disponible es de *48* Bytes y expandible a 2 Kb; la velocidad de comunicación es de 106 Kbps.

- Etiqueta Tipo 3: Se basa en la *Japanese Industrial Standard (JIS) X 6319-4*, también conocido como FeliCa. Se encuentran ya configuradas desde la fabricación, ya sea para escritura, o sólo para lectura. La memoria disponible es variable, pero teóricamente el límite de memoria es 1 MB por servicio; la velocidad de comunicación es de 212 Kbps o 424 Kbps.

- Etiqueta Tipo 4: Es plenamente compatible con ISO14443A y los estándares B. Se encuentran ya configuradas desde la fabricación, ya sea para lectura y escritura, o de sólo lectura. La memoria disponible es variable, de hasta 32 KB por servicio; la velocidad de comunicación es de hasta 424 Kbps.

	Tipo 1	Tipo 2	Tipo 3	Tipo 4
Interfaz RF	ISO 14443 A-2	ISO 14443 A-2	FeliCa (ISO 18092, modo de comunicación pasivo a 212 kbits/sec)	ISO 14443 A-2
Inicialización	ISO 14443 A-3	ISO 14443 A-3	FeliCa (ISO 18092, modo de comunicación pasivo a 212 kbits/sec)	ISO 14443 A-3
Velocidad	106 kbits/sec	106 kbits/sec	212 kbits/sec	106-424 kbits/sec
Protocolo	Conjunto de comandos específicos	Conjunto de comandos específicos	Protocolo FeliCa	Comandos ISO 14443 A-4 ISO 7816-4
Tamaño de memoria	Mayor de 1KB	Mayor de 2 KB	Mayor de 1 MB	Mayor de 64 KB
Coste (dependiendo de la memoria)	Bajo	Bajo	Moderado	Moderado
Casos de uso	Etiquetas con tamaño de memoria pequeño para aplicaciones sencillas		Etiquetas flexibles con tamaño de memoria grande para aplicaciones con múltiples capacidades	

Tabla 1. Relación de los distintos tipos de etiquetas y sus características

2.1.3.7 Arquitectura de un dispositivo móvil NFC

En este apartado se define la arquitectura de un dispositivo móvil NFC, así como las distintas opciones de ubicación del Elemento Seguro, relacionándolo con el uso de una aplicación de pago, ya que se trata la aplicación realizada en este proyecto.

Los principales componentes de un dispositivo móvil NFC son una bobina o antena incorporada en el interior del teléfono, el chip NFC y el denominado elemento seguro, que es un chip con características de seguridad similares a las encontradas en las tarjetas inteligentes y que se encarga de procesar de forma segura las transacciones, al contener en su interior la información bancaria del cliente. *[6]*

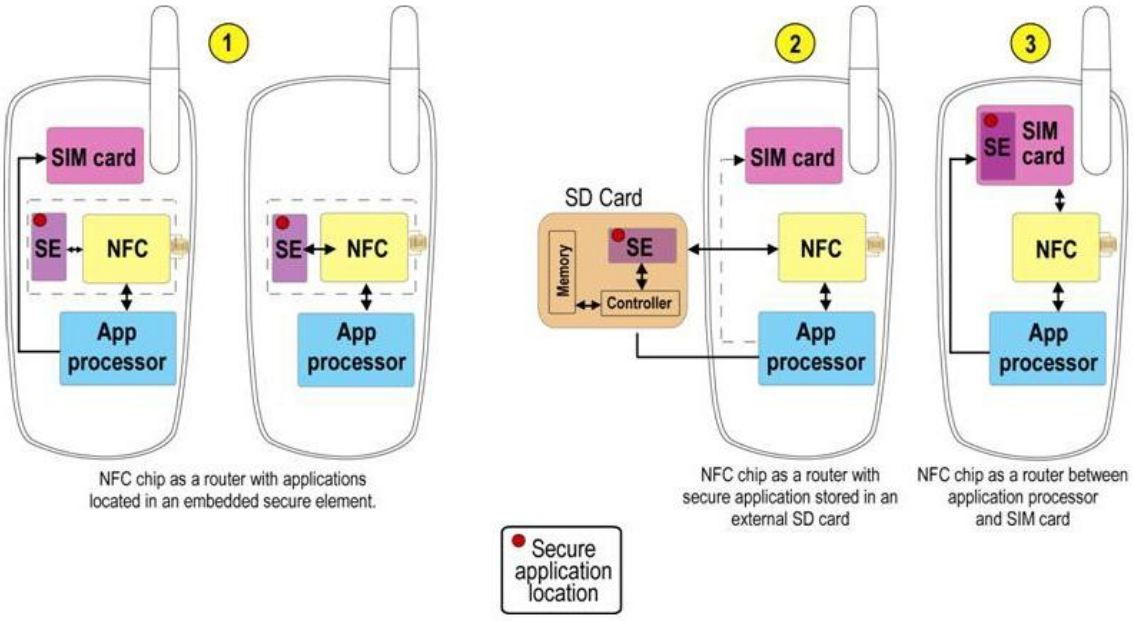

Figura 14: Esquema del interior de un teléfono NFC, según la situación del elemento seguro.

El elemento seguro es uno de los componentes más importantes en la arquitectura hardware del dispositivo NFC y juega un papel determinante en la definición de la arquitectura NFC. Los operadores de telefonía móvil junto con las instituciones financieras juegan un papel esencial en la selección del elemento seguro más adecuado, para las aplicaciones de pago.

Actualmente, existen tres opciones para la ubicación del elemento seguro dentro del teléfono móvil NFC, como se puede observar en Figura 14:

1. Elemento seguro incorporado en la electrónica del móvil: El elemento seguro puede ser un microcontrolador incorporado en el dispositivo móvil, bien montado en la placa base directamente o conectado de alguna forma a la placa base. Esta es la arquitectura que ha sido utilizada más habitualmente en los diversos proyectos pilotos en todo el mundo. La principal ventaja de esta aproximación es que el elemento seguro que puede ser incorporado en la actualidad tiene todas las certificaciones necesarias hardware y software demandadas por el sector bancario. Sin embargo, la principal desventaja de esta solución es su incapacidad de gestionar las credenciales de pago del usuario cuando éste quiere cambiar de teléfono móvil.

2. Tarjeta de memoria como elemento seguro: con esta solución, una tarjeta de memoria (SD, MMC, MS, etc) incorpora un chip seguro con un

microcontrolador y memoria flash. Esta solución permite que terceras partes puedan suministrar tarjetas precargadas con su aplicación.

3. Tarjeta SIM como elemento seguro: con esta arquitectura, la tarjeta SIM incorpora la aplicación de pago, así como cualquier otra aplicación NFC. La aplicación puede almacenarse en el propio SIM o como un componente adicional en el conector del SIM. Esta es la solución que más interesa a las operadoras móviles porque estaría a su cargo la gestión de la información.

2.1.3.8 Seguridad en NFC

NFC por sí sola no asegura comunicaciones seguras, no ofrece protección contra los que se dedican a escuchar comunicaciones y es también vulnerable a modificación de datos. Las aplicaciones deben usar protocolos criptográficos para establecer un canal seguro.

Sin embargo, esto se contrarresta con la distancia de operación necesaria de NFC, ya que las transacciones sólo se pueden activar en un rango de acción muy limitado lo que limita seriamente el uso de la tecnología sin conocimiento del usuario.

La sencillez y simplicidad, por tanto, de la interacción “aproximación del dispositivo” lleva inherentemente las características de confianza y seguridad.

Además, al ser de corto alcance, evita los errores de comunicación, asegura una mayor eficacia en la transmisión de datos y hace que el posible deba estar dentro de ese corto rango, en el que el usuario podría darse cuenta fácilmente.

2.1.4 Aplicaciones y casos de uso

2.1.4.1 Introducción

Actualmente, los usos de NFC están ligados a los teléfonos móviles, debido a su ubicuidad y al hecho de que sea el único dispositivo que todos necesitamos llevar a todas partes.

Se están llevando a cabo en los últimos años múltiples pruebas piloto en las que poner a prueba todas las ideas de aplicaciones que pueden llevarse a cabo a través de esta tecnología.

Los estudios llevados a cabo por el NFC Forum señalan que en un periodo de tres a cinco años, la tercera parte de los teléfonos móviles a nivel mundial estén equipados con tecnología NFC. Esto, entre otras muchas cosas, nos permitirá:

- Hacer pagos con tan sólo acercar el teléfono a un punto terminal de venta o una máquina expendedora.
- Obtener información, cupones, descuentos y ofertas de póster inteligentes que cuentan con una etiqueta NFC.
- Almacenar y recoger entradas para control de acceso en garajes, cines, conciertos o cualquier otro tipo de evento.
- Almacenar información personal que posibilite el acceso seguro a edificios.
- Sacar una fotografía y transmitirla de forma inalámbrica a cualquier televisor o impresora sin necesidad de realizar ninguna configuración.
- Compartir tarjetas de visita y contacto con otros teléfonos NFC.

2.1.4.2 Tipos de aplicaciones

Formalmente, el NFC Forum ha especificado tres aplicaciones básicas centrales:

- Conexión NFC: Transferencia de datos entre dispositivos. Ejemplos:
  - Establecer las conexiones *wireless* de la oficina o de una casa.
  - Acceso a edificios o a eventos.
  - Tocar una impresora para imprimir.
- Acceso NFC: Acceder a información "*on-the-move*" (en movimiento). Ejemplos:
  - Leer posters inteligentes.
  - Compartir tarjetas de contacto.
- Transacciones NFC: Pago a través de móviles y tickets. Aplicaciones seguras. Ejemplos:
  - Pagar para bienes y servicios.
  - Comprar tickets de viaje, y acceso a trenes, aviones, etc.

Figura 15: Ejemplo de uso de NFC en Japón: para pagar en una máquina expendedora y para acceder al metro

2.1.4.3 La idea de "*mobile wallet*"

Se entiende que la tecnología NFC puede en realidad hacer mucho más fácil las tareas del día a día: realizar pagos, uso del transporte público, manejo de identificaciones, compartir información, etc. En los dispositivos móviles NFC se podrá tener lo que se conoce como "Billetera electrónica", donde se sustituirá un gran porcentaje de las tarjetas de crédito e identificaciones, además de otro gran número de objetos.

En la Figura 16 se puede ver uno de los casos de uso desarrollado por Google, ejemplo de las muchas aplicaciones que se le puede dar a un móvil con tecnología NFC. Se trata de Google *HotPot*, un sistema para poder hacer recomendaciones de lugares específicos que puedan tener interés. Acercando el móvil NFC a una pegatina NFC localizada en un lugar específico, se puede hacer recomendaciones del mismo de forma instantánea.

Figura 16: Ejemplo del teléfono Nexus S, acercándose a una pegatina NFC *Hotpot*

A continuación, en la siguiente Tabla 2, se puede observar distintos casos de uso en distintos lugares:

Station Airport	Paso por la puerta de embarque	Recoger información de un poster inteligente	Recoger información de un kiosco	Pago del taxi/bus
Vehicle	Personalizar la posición del asiento	Uso de representación del carnet de conducir	Pagar el coste de parking	Abrir el automóvil
Office	Entrar/salir de la oficina	Intercambiar tarjetas de contacto de negocio	Identificarse en un PC	Imprimir información usando una impresora
Store Restaurant	Pagar usando una tarjeta de crédito	Acumular puntos de fidelización	Recoger y usar cupones de descuento	Compartir información y cupones entre usuarios
Theatre Stadium	Recoger información de eventos	Pagar un ticket	Pasar por la entrada	
Anywhere	Descargar y personalizar aplicaciones	Descargarse tickets	Aplicaciones de Salud/Hospitales	Realizar pagos

Tabla 2: Ejemplos de usos de la tecnología NFC en distintos entornos

2.1.4.4 Pago a través del móvil

El caso de uso en el que se centra este proyecto, trata de poder pagar con tu móvil en cualquier establecimiento y para cualquier bien o servicio, en vez de con tu dinero físico o con tu tarjeta de crédito/debido, con el simple hecho de tocar un lector u otro dispositivo con tu móvil.

Sin duda, es uno de los casos que más se están probando y para el que se están realizando más pruebas piloto, más conferencias, o creándose más estándares.

Figura 17: Ilustración de un pago a través de un móvil NFC en un terminal fijo Visa

En este apartado, se da una visión del estado actual del pago a través de un móvil con la tecnología NFC en el mundo:

- **Dispositivos NFC en el mercado:**
  En el mercado ya existen o están a punto de salir a la venta, dispositivos móviles *smartphones* con capacidades NFC. Ejemplos:
  - Google: Nexus S, desarrollado también por Samsung, ha salido recientemente al mercado.
  - Blackberry: durante el año 2011 planea sacar al mercado 5 dispositivos, todos ellos con capacidad NFC.

- **Fabricantes contra operadoras móviles:**
  Existe un frente de batalla claro entre los operadores móviles y los fabricantes de dispositivos y grandes compañías (Apple, Google, Blackberry). Esto es debido al tema tratado en el apartado 2.1.3.7, en el que se habla de la localización del Elemento Seguro dentro del teléfono móvil. Si éste es localizado en la SIM, el poder de decisión lo tendrá las operadoras móviles. Si está localizado en la electrónica del móvil, los fabricantes de dispositivos serán los que ejerzan el control sobre él.

- **Las operadoras móviles lanzan estándares de pago:**
  Se ha firmado recientemente un acuerdo estándar NFC España entre Telefónica, Vodafone, Orange. El acuerdo está abierto a otros operadores que deseen adherirse al mismo y rechaza cualquier plan comercial conjunto, ya que cada operadora tendrá libertad para elegir la comercialización de esta iniciativa.
  En Reino Unido, ya se ha constatado un acuerdo para pagos a través del móvil entre las operadoras Vodafone, Orange, O2 y otras, declarando que lanzarán servicios de pago en el año 2012.
  En Estados Unidos, se ha creado “Isis”, una unión *Joint Venture* de varias operadoras móviles estadounidenses, para crear una red comercial móvil usando dispositivos móviles *smartphones* y NFC.

- **Las grandes compañías de móviles realizan pruebas piloto:**
  Ejemplos de ellas son:
  - Google: realización de una prueba piloto a mediados del 2011 para el pago con móviles a través de NFC en Nueva York y San Francisco, utilizando como sistema de pago su sistema propietario *Google Checkout*.
  - Blackberry: realización de una prueba piloto de una aplicación propietaria Blackberry como sistema de pago, en asociación con Bank of America y Mastercard.

- **Las entidades bancarias también preparan sus sistemas financieros:**
  Ejemplo de ello son los estándares de pago *PayPass* de MasterCard o *Visa Wave* de Visa.

2.2 La tecnología Bluetooth

2.2.1 Introducción

Es una tecnología desarrollada por Ericsson en 1994, que hace factible la conectividad inalámbrica entre dispositivos a corta distancia. Éstos pueden llegar a formar redes con diversos equipos de comunicación: móviles, radiolocalizadores, e inclusive, electrodomésticos.

El estándar Bluetooth se compone de dos capítulos, uno de ellos describe las especificaciones técnicas principales, mientras que el otro define perfiles específicos para aplicaciones, que aseguran la interoperabilidad de dispositivos Bluetooth entre fabricantes.

El *Bluetooth Special Interest Group* (SIG), es una asociación comercial formada por líderes en telecomunicaciones, que está conduciendo el desarrollo de la tecnología inalámbrica Bluetooth y llevándola al mercado.

Bluetooth es una tecnología de ondas de radio de corto alcance (2.4 gigahertzios de frecuencia) cuyo objetivo es el de simplificar las comunicaciones o la sincronización de datos entre dispositivos. Permite comunicaciones, incluso a través de obstáculos, a distancias de hasta unos 10 metros.

Entre los promotores de Bluetooth, se encuentran Agere, Ericsson, IBM, Intel, Microsoft, Motorola, Nokia y Toshiba, etc.

Los objetivos principales de la tecnología Bluetooth son:

- Permitir la comunicación sencilla entre dispositivos fijos y móviles.
- Evitar la dependencia de cables que permitan la comunicación.
- Permitir la creación de pequeñas redes de forma inalámbrica.

2.2.2 Origen del nombre Bluetooth

El nombre proviene de Harald Blåtand, cuya traducción al inglés sería Harold Bluetooth, un Vikingo y rey de Dinamarca, de los años 940 a 981. Fue reconocido por su capacidad de ayudar a la gente a comunicarse. Durante su reinado unió Dinamarca y Noruega.

El logo de Bluetooth fusiona las runas germánicas análogas a las letras latinas modernas H y B: ᚼ(Hagall) y ᛒ(Berkanan), unificadas en una runa.

Figura 18: Logo de la tecnología Bluetooth

2.2.3 Funcionamiento

Trabaja en dos capas del modelo OSI, que son la de enlace y aplicación, incluye un transpondedor que trasmite y recibe a una frecuencia de 2.4 GHz. Las conexiones que se realizan son de uno a uno con un rango máximo de 10 metros. Si se desea aumentar la distancia, se tendría que utilizar repetidores, los cuales ayudarían a abarcar una distancia de 100 metros.

Bluetooth, por cuestiones de seguridad, cuenta con mecanismos de encriptación de 64 bits y autentificación, para controlar la conexión y evitar que dispositivos puedan acceder a los datos o realizar modificaciones.

Durante la transferencia de datos el canal de comunicaciones permanece abierto y no requiere la intervención directa del usuario cada vez que se desea transferir voz o datos de un dispositivo a otro. *[7], [8]*

2.2.4 Casos de uso

Las posibilidades de uso de la tecnología Bluetooth son múltiples. Algunas de las posibilidades actuales son:

- Eliminación de la necesidad de conexiones por cable entre los productos y accesorios electrónicos.
- Intercambio de archivos, tarjetas de visita, citas del calendario, etc. entre usuarios de Bluetooth.
- Sincronización y transferencia de archivos entre dispositivos.
- Conexión a determinados contenidos en áreas públicas.

Figura 19: Ejemplos de uso de Bluetooth

2.2.5 NFC y Bluetooth

NFC y Bluetooth son dos tecnologías de comunicación de corto alcance que podemos encontrar integradas en los teléfonos móviles.

Para evitar un proceso de configuración complicado, NFC puede utilizarse para iniciar tecnologías inalámbricas, como Bluetooth. Con NFC, en lugar de realizar una configuración manual para identificar dispositivos, la conexión entre dos dispositivos, es automática (en menos de una décima de segundo).

La tasa de transferencia máxima de NFC (424 kbit/s), es menor que en la de Bluetooth v2.1 (2.1 Mbit/s). El rango de trabajo en NFC es menor a 20cm, lo que reduce la probabilidad de intercepciones fraudulentas. Esto hace NFC interesante en áreas donde transmitir y usar una señal física sea difícil. En contraste con Bluetooth, NFC es compatible con infraestructuras RFID y requiere menor potencia.

Estas y otras características de estas tecnologías, se recogen en la Tabla 3 en forma de comparativa.

	NFC	Bluetooth
Compatibilidad RFID	ISO 18000-3	no
Estándar	ISO/IEC	Bluetooth SIG
Estándar de red	ISO 13157, etc.	IEEE 802.15.1
Tipo de red	Punto a punto	Punto a multipunto
Rango	~ 10 cm	~10 m (clase 2) ~100m (clase 3)
Frecuencia	13.56 MHz	2.4-2.5 GHz
Tasa de transferencia	424 Kbit/s	2.1 Mbit/s (v2.1) (mayor a 721 Kbit/s)
Tiempo de inicialización	< 0.1 s	~ 6 s
Seguridad	Sí, a nivel hardware y protocolo	Sí, a nivel protocolo
Modos de comunicación	Activo-pasivo Activo-activo	Activo-activo

Tabla 3: Comparativa entre las tecnologías NFC y Bluetooth

Además, es interesante añadir que el estándar Bluetooth 2.1 incorpora *"NFC Cooperation*". Se trata de la creación automática de conexiones Bluetooth seguras cuando una interfaz NFC se encuentre disponible. Por ejemplo:

- Unos auriculares con Bluetooth 2.1 pueden conectarse a un móvil con tecnología NFC simplemente acercando los dispositivos.
- Se pueden enviar fotos de un móvil o una cámara de fotos a un marco digital simplemente acercando el teléfono o la cámara al marco.

2.3 Java

Java es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90.

El lenguaje se asemeja a la sintaxis de C y C++, pero su modelo de objetos es más simple y elimina herramientas de bajo nivel, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un *bytecode*, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el *bytecode* es normalmente interpretado o compilado a código nativo para la ejecución.

La implementación original y de referencia del compilador, la máquina virtual y las bibliotecas de clases de Java fueron desarrolladas por Sun Microsystems en 1995.

Desde entonces, Sun ha controlado las especificaciones, desarrollo y evolución del lenguaje a través del *Java Community Process*. También se han desarrollado implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

En 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del *Java Community Process,* de tal forma que prácticamente todo el Java de Sun es actualmente software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java aún no lo es*).[9]*

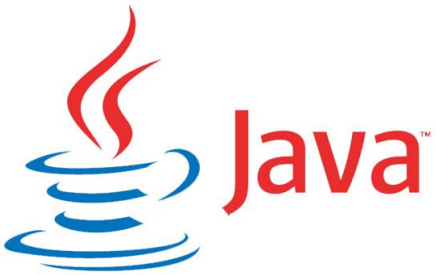

Figura 20: Logo de Java

2.3.1 Características principales de Java

- Simple: Java ofrece toda la funcionalidad de un lenguaje potente, pero sin las características menos usadas y más confusas de éstos.

- Orientado a objetos: Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos que manipulan esos datos.

- Distribuido: Proporciona una colección de clases para su uso en aplicaciones de red distribuidas.

- Robusto: Java fue diseñado para crear software altamente fiable. Para ello proporciona numerosas comprobaciones en compilación y en tiempo de ejecución.

- Indiferente a la arquitectura: El compilador de Java genera *bytecode*, un formato intermedio indiferente a la arquitectura, diseñado para transportar el código eficientemente a múltiples plataformas hardware y software.

- Portable: La indiferencia a la arquitectura representa sólo una parte de su portabilidad. Además, Java especifica los tamaños de sus tipos de datos

básicos y el comportamiento de sus operadores aritméticos, de manera que los programas son iguales en todas las plataformas.

- Multihilo: Java soporta sincronización de múltiples hilos de ejecución (*multithreading*) *[10]*

2.3.2 Plataformas y APIs

Sun define tres plataformas para cubrir distintos entornos de aplicación:

- Java ME (*Java Platform, Micro Edition*) o J2ME: orientada a entornos de limitados recursos, como teléfonos móviles, PDAs, etc.
- Java SE (*Java Platform, Standard Edition*) o J2SE: para entornos de gama media y estaciones de trabajo. Aquí se sitúa al usuario medio en un PC de escritorio.
- Java EE (*Java Platform, Enterprise Edition)* o J2EE: orientada a entornos distribuidos empresariales o de Internet.

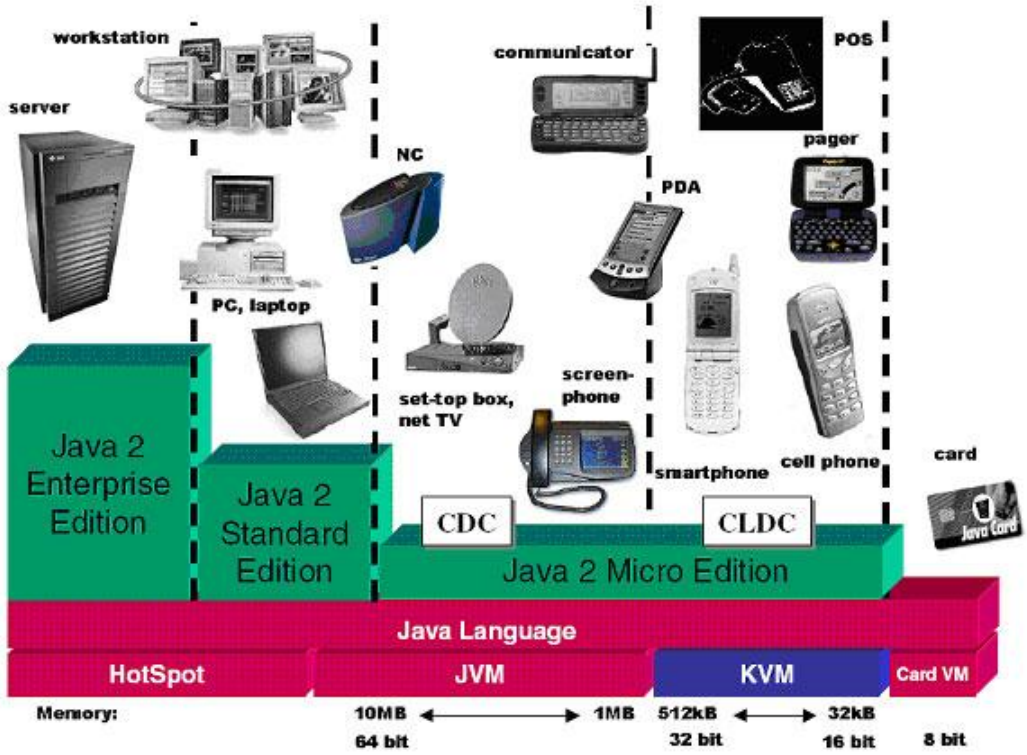

Figura 21: Diagrama de las distintas plataformas Java, sus configuraciones y a qué dispositivos están orientadas

Las clases en las APIs (interfaz de aplicaciones programables) de Java se organizan en grupos disjuntos llamados paquetes. Cada paquete contiene un conjunto de interfaces, clases y excepciones relacionadas. *[11]*

2.3.3 JRE, SDK o JDK

El JRE (*Java Runtime Environment*, o Entorno en Tiempo de Ejecución de Java) es el software necesario para ejecutar cualquier aplicación desarrollada para la plataforma Java. El usuario final usa el JRE como parte de paquetes software o *plugins* (conectores) en un navegador Web. Sun ofrece también el SDK de Java, o JDK (*Java Development Kit*) en cuyo seno reside el JRE, e incluye herramientas como el compilador de Java, *Javadoc* para generar documentación o el depurador.

2.3.4 IDE

Mientras un JDK/SDK ofrece las herramientas para compilar y ejecutar programas en Java, éste no ofrece un ambiente de trabajo para proyectos complejos.

Los IDE's (*Integrated Development Environment*) ofrecen un ambiente gráfico en los que se tiene acceso a mayor número de herramientas no ofrecidas en los JDK's: Depuradores más elaborados, *check-points* dentro de la compilación, creación de WAR's (*Web-Archives*), entre otras cosas.

El IDE utilizado para realización de este proyecto ha sido NetBeans.

2.3.4.1 NetBeans

NetBeans es un proyecto de código abierto con una gran base de usuarios. Sun MicroSystems fundó el proyecto de código abierto NetBeans en Junio 2000.

NetBeans IDE es un entorno de desarrollo, una herramienta para que los programadores puedan escribir, compilar, depurar y ejecutar programas.

Está escrito en Java, pero puede servir para cualquier otro lenguaje de programación. Existe además un número importante de módulos para extenderlo.

2.4 J2ME

2.4.1 Introducción

J2ME es la edición de Java enfocada a la aplicación de la tecnología en dispositivos electrónicos con capacidades computacionales y gráficas muy reducidas, tales como teléfonos móviles, PDAs o electrodomésticos inteligentes.

Tiene unos componentes básicos que la diferencian de las otras versiones, como el uso de una máquina virtual denominada KVM (*Kilo Virtual Machine,* debido a que requiere sólo unos pocos Kilobytes de memoria para funcionar) en vez del uso de la JVM (Máquina Virtual de Java) clásica o inclusión de un pequeño y rápido recolector de basura.

J2ME representa una versión simplificada de J2SE. Sun separó estas dos versiones ya que J2ME estaba pensada para dispositivos con limitaciones de proceso y capacidad gráfica. También separó J2SE de J2EE porque este último exigía unas características muy pesadas o especializadas de E/S y trabajo en red.

Hoy, J2EE es un superconjunto de J2SE pues contiene toda la funcionalidad de éste y más características, así como J2ME es un subconjunto de J2SE (excepto por el paquete javax.microedition) ya que, como se ha mencionado, contiene varias limitaciones con respecto a J2SE. *[12], [13], [14]*

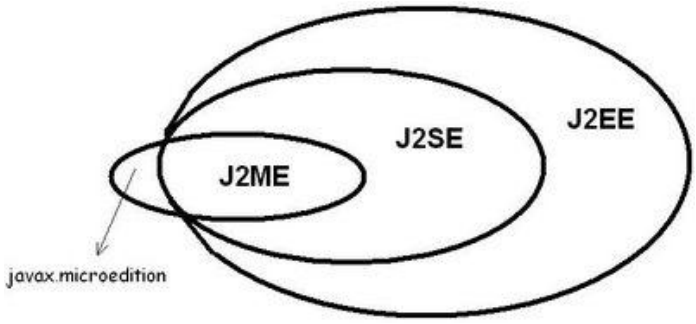

Figura 22: Esquema de los conjuntos y herencias de las distintas versiones de Java

2.4.2 Descripción general y configuración

Los componentes que forman parte de esta tecnología, y que conforman un entorno de ejecución determinado de J2ME, son:

- Máquinas virtuales: con diferentes requisitos, dependiendo del tipo de dispositivo.
- Configuraciones: conjunto de clases básicas orientadas a conformar el corazón de las implementaciones para dispositivos de características específicas:

    - *Connected Limited Device Configuration* (CLDC): enfocada a dispositivos con restricciones de procesamiento y memoria.
    - *Connected Device Configuration* (CDC): enfocada a dispositivos con más recursos.

- Perfiles: bibliotecas Java de clases específicas orientadas a implementar funcionalidades de más alto nivel para familias específicas de dispositivos.
- Paquetes opcionales:conjunto de APIs de una tecnología específica

<table>
<tr><td rowspan="2"></td><td colspan="4">Aplicación</td></tr>
<tr><td rowspan="2">Perfiles</td><td>RMI Profile</td><td>Personal Profile</td><td rowspan="2">PDA Profile</td><td rowspan="2">MID Profile</td></tr>
<tr><td colspan="2">Foundation Profile</td></tr>
<tr><td>Configuraciones</td><td colspan="2">CDC</td><td colspan="2">CLDC</td></tr>
<tr><td>Máquina Virtual Java</td><td colspan="2">CVM</td><td colspan="2">KVM</td></tr>
<tr><td rowspan="2"></td><td colspan="4">Sistema Operativo</td></tr>
<tr><td colspan="4">Dispositivos Hardware</td></tr>
</table>

Figura 23: Esquema de configuración de un entorno de ejecución en J2ME

2.4.2.1 Configuraciones

Una configuración es el conjunto mínimo de APIs Java que permiten desarrollar aplicaciones para un grupo de dispositivos. Éstas APIs describen las características básicas, comunes a todos los dispositivos.

2.4.2.1.1 CLDC (Connected Limited Device Configuration)

Está orientado hacia pequeños dispositivos, como teléfonos móviles o PDAs.

2.4.2.1.2 CDC (Connected Device Configuration).

Está orientado hacia dispositivos de mayor capacidad que los anteriores, como televisiones con Internet, algunos electrodomésticos y sistemas de navegación en vehículos.

2.4.2.2 Máquinas virtuales

Una máquina virtual de Java (JVM) es un programa encargado de interpretar código intermedio (*bytecode*) de los programas Java precompilados a código máquina ejecutable por la plataforma, efectuar las llamadas pertinentes al sistema operativo subyacente y observar las reglas de seguridad y corrección de código definidas para el lenguaje Java.

Las configuraciones CLDC y CDC definen un conjunto propio de características soportadas de la Máquina Virtual de Java. Como consecuencia, cada una requiere su propia máquina:

2.4.2.2.1 Kilo Virtual Machine (KVM)

La máquina virtual de la configuración CLDC se denomina *Kilo Virtual Machine (*KVM). Se trata de una implementación de Máquina Virtual reducida y especialmente orientada a dispositivos con bajas capacidades computacionales y de memoria.

2.4.2.2.2 Compact Virtual Machine (CVM).

La máquina virtual de la configuración CDC se denomina CVM (*Compact Virtual Machine*). Soporta las mismas características que la Máquina Virtual de J2SE.

2.4.2.3 Perfiles

El perfil es la parte que define las APIs que controlan el ciclo de vida de la aplicación, interfaz de usuario, etc.

Establece unas APIs que definen las características de un dispositivo, mientras que la configuración hace lo propio con una familia de ellos. Esto hace que a la hora de construir una aplicación se cuente tanto con las APIs del perfil como de la configuración.

Existen unos perfiles que se construyen sobre la configuración CDC y otros sobre la CLDC:

- Perfiles para la configuración CDC:
  - *Foundation Profile*.
  - *Personal Profile*.
  - *RMI Profile*.
- Perfiles para la configuración CLDC:
  - *PDA Profile*.

- *Mobile Information Device Profile* (MIDP).

2.4.2.3.1 Mobile Information Device Profile (MIDP).

Este perfil está construido sobre la configuración CLDC. Los tipos de dispositivos que se adaptan a estas características son: teléfonos móviles, buscapersonas (*pagers*) o PDAs de gama baja con conectividad.

El perfil MIDP establece las capacidades del dispositivo, por lo tanto, especifica las APIs relacionadas con:

- La aplicación (semántica y control de la aplicación MIDP).
- Interfaz de usuario.
- Almacenamiento persistente.
- Trabajo en red.
- Temporizadores.

Las aplicaciones que se realizan utilizando MIDP reciben el nombre de *MIDlets.*

2.4.2.4 Paquetes opcionales

En la siguiente Tabla 4 se detallan los distintos paquetes opcionales que existen para J2ME.

Paquete opcional	Perfiles	Descripción
Wireless Messaging API (WMA), JSR120, JSR205.	CLDC, MIDP	Mensajería sin cables. Acceso a recursos de comunicación sin cable como la mensajería SMS
Information Module Profile (IMP), JSR 195.	CLDC, MIDP	Proporciona un entorno de aplicación para dispositivos embebidos que no tiene grandes capacidades gráficas o con recursos limitados.
Mobile Media API (MMAPI), JSR 135.	CLDC, MIDP	Soporte de audio, video y otros tipos de datos multimedia basados en tiempo para dispositivos de recursos limitados.
Location API J2ME, JSR-179.	CLDC 1.1, CDC	Esta especificación permite la localización de dispositivos móviles para dispositivos con recursos limitados
SIP API para J2ME, JSR-180	CLDC	El protocolo Session Initiation Protocol (SIP) se utiliza para establecer y gestionar sesiones IP multimedia.
Contactless Communication API, JSR-257	CLDC	Proporciona interfaz para comunicaciones sin contacto
Security and Trust Services API J2ME (SATSA), JSR-177	CLDC	Amplía las características de seguridad para la plataforma J2ME

Mobile 3D Graphics (M3G), JSR-184	CLDC, MIDP	permite generar gráficos tridimensionales a frecuencias de imagen interactivas en dispositivos de recursos restringidos
Web Services APIs (WSA), JSR 172	CLDC, MIDP (1.0, 2.0)	Amplía la plataforma de servicios web para incluir J2ME.
Bluetooth API, JSR-82	CLDC, MIDP (1.0, 2.0)	Proporciona un estándar para la creación de aplicaciones Bluetooth
RMI Optional Package (RMI OP), JSR 66	CDC	Permite a dispositivos de consumo y aplicaciones embebidas interactuar como aplicaciones distribuidas.
JDBC API, JSR-169	CDC	Define un subconjunto del API JDBC 3.0 que se puede utilizar para procesar BBDD

Tabla 4: Relación de los paquetes opcionales para J2ME

2.4.3 MIDlet

Un *MIDlet* es una aplicación Java realizada con el perfil MIDP sobre la configuración CLDC, utilizando la máquina virtual KVM. Esta será la configuración utilizada en este proyecto al realizar aplicaciones sobre un dispositivo móvil Nokia.

Siempre hereda de la clase *javax.microedition.midlet.MIDlet*. Los métodos de esta clase permiten a la aplicación crear, empezar, parar y destruir un *Midlet*.

2.4.3.1 Ciclo de vida de un MIDlet

El ciclo de vida de un *Midlet* se compone de los siguientes estados: Pausado, Activo o Destruido. Sólo puede estar en un estado a la vez. En la Figura 24 se muestra como se pasa de un estado a otro.

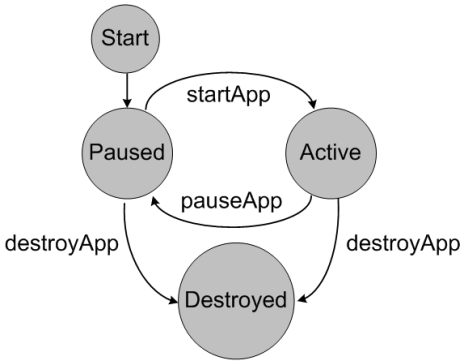

Figura 24: Ciclo de vida de un Midlet

2.4.3.2 Empaquetamiento de un MIDlet

Los *Midlet* necesitan ser empaquetados antes de ser instalados en el dispositivo de destino. Se utiliza un fichero JAR (Java ARchive) donde se tiene el *Midlet* principal y todas aquellas clases, imágenes u otros ficheros que puedan ser necesarios en tiempo de ejecución. Un fichero JAR puede contener varios Midlets, a esto se le denomina *Midlet suite* y permite compartir recursos.

También se incluirá en el JAR información (en el fichero *manifest*) que explica al dispositivo el contenido del fichero JAR. Esta misma información también se incluye en el fichero JAD (*Java Application Descriptor*).

2.5 Java Swing

2.5.1 Introducción

Una interfaz gráfica de usuario (GUI) es la parte del programa que permite al usuario interaccionar con él. Ofrece al usuario ventanas, cuadros de diálogo, barras de herramientas, botones, listas desplegables y otros elementos.
Las aplicaciones son conducidas por eventos y se desarrollan haciendo uso de las clases que para ello ofrece la API de Java.

El funcionamiento del modelo de eventos se basa en la gestión de excepciones.
Para cada objeto que represente una interfaz gráfica, se pueden definir objetos "oyentes" (*Listeners*), que esperen a que suceda un determinado evento sobre la interfaz. Por ejemplo se puede crear un objeto oyente que esté a la espera de que

el usuario pulse sobre un botón de la interfaz, y si esto sucede, él es avisado, ejecutando determinada acción.

Cuando apareció Java, los componentes gráficos disponibles para el desarrollo de GUI se encontraban en una biblioteca conocida como *Abstract Window Toolkit* (AWT). AWT es adecuada para interfaces gráficas sencillas, pero no para proyectos más complejos.

Swing es una de las mejoras principales que ha experimentado el JDK en su versión 1.2 con respecto a la versión 1.1, y representa la nueva generación de AWT. Es una de las API de las Clases de Fundamentos de Java (JFC), lo cual es el resultado de un esfuerzo de colaboración entre Sun, Netscape, IBM y otras empresas. *[15], [16]*

2.5.2 JFC

JFC es la abreviatura de *Java Foundation Classes*, que comprende un grupo de características para ayudar a construir interfaces gráficos de usuario (GUIs), recogidas en la Tabla 5.

Las JFC contienen dos paquetes gráficos: AWT y Swing.
AWT presenta componentes pesados, que en cada plataforma sólo pueden tener una representación determinada. Está disponible desde la versión 1.1 del JDK como *java.awt*.
Swing presenta componentes ligeros, que pueden tomar diferente aspecto y comportamiento. Está disponible desde la versión 1.2 del JDK como *javax.swing*.

Función	Descripción
Componentes Swing GUI	Incluye todo desde botones hasta tablas.
Soporte conectable *Look-and-Feel* (Aspecto y Comportamiento)	Le ofrece a cualquier componente Swing una amplia selección de aspectos y comportamientos. Por ejemplo, el mismo programa puede usar el *Look&Feel* Java o Windows
Accesibilidad API	Permite tecnologías asistivas como lectores de pantalla y *display* Braille para obtener información desde el interface de usuario
Java 2D API	Permite a los desarrolladores incorporar fácilmente gráficos 2D de alta calidad, texto, e imágenes en aplicaciones y *applets* Java.
Internacionalización	Permite a los desarrolladores construir aplicaciones que pueden interactuar con usuarios de todo el mundo en sus propios idiomas.

Tabla 5: Características gráficas definidas por lFC

2.5.3 Características principales de Swing

Swing es un paquete de Java para la generación del GUI en aplicaciones reales de gran tamaño.

Los componentes esenciales de Swing se contemplan en la Figura 25, donde las clases AWT están en amarillo, y las clases de Swing están en azul. Los componentes Swing tienen nombres similares a los componentes AWT, pero empiezan por J.

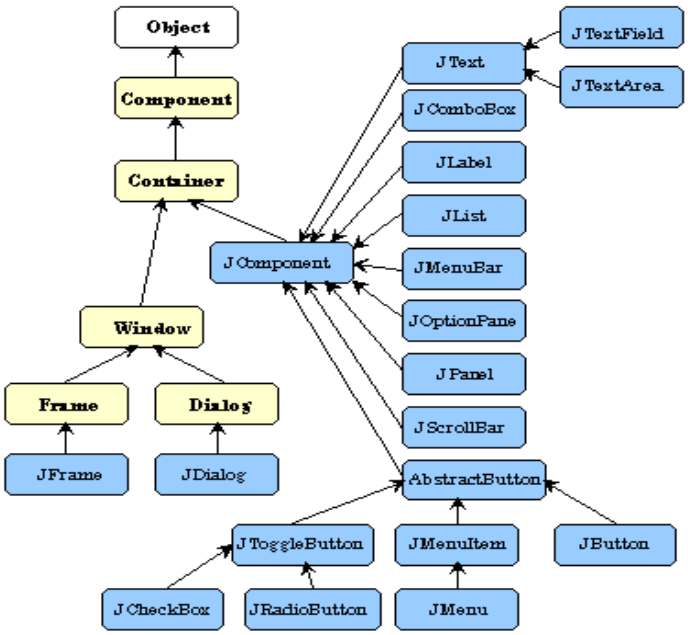

Figura 25: Jerarquía de Componentes Swing

Los componentes son los elementos básicos de la programación con Swing. Todo lo que se ve en un GUI de Java es un componente. Los componentes se colocan en otros elementos llamados contenedores que sirven para agrupar componentes.

La clase javax.swing.JComponent es la clase padre de todos los componentes. A su vez, JComponent desciende de java.awt.container y ésta de java.awt.component.

Swing posee algunos contenedores especiales. Algunos son:

- JWindow: Representa un panel de ventana sin bordes ni elementos visibles.
- JFrame: Objeto que representa una ventana típica con bordes, botones de cerrar, etc.

- JPanel: Es la clase utilizada como contenedor genérico para agrupar componentes.
- JDialog: Clase que genera un cuadro de diálogo.
- JApplet: Contenedor que agrupa componentes que serán mostrados en un navegador.

En la siguiente Figura 26, podemos observar algunos ejemplos gráficos de componentes básicos en su Look&Feel de Java:

Figura 26: Ejemplos gráficos de JButton, JCheckBox, JComboBox y JList

2.6 Tecnologías en el lado del servidor

Para la consecución de la aplicación web realizada, se han utilizado distintas tecnologías, que se detallan a continuación.

Una aplicación web es un conjunto de *servlets*, *JSP*s y otros recursos (ficheros HTML, imágenes, ficheros de configuración, etc.) relacionados entre sí por formar parte de la misma aplicación.

Estos recursos se pueden empaquetar y ejecutar en un contenedor web en un servidor para ofrecer una determinada funcionalidad a la que los clientes acceden típicamente a través de un navegador. Este servicio se ha desarrollado con Apache Tomcat.

Además, se ha requerido el despliegue de una base de datos, que se ha desarrollado a través de JavaDB Apache Derby

2.6.1 Java Servlets

Un *servlet* es un programa Java que se ejecuta en un servidor (normalmente HTTP) y extiende su funcionalidad. Atiende peticiones recibidas desde los clientes y genera las respuestas.

Los *servlets* son módulos que extienden los servidores orientados a petición-respuesta, como los servidores web compatibles con Java. Por ejemplo, un *servlet* podría ser responsable de tomar los datos de un formulario de entrada de pedidos en HTML y aplicarle la lógica de negocios utilizada para actualizar la base de datos de pedidos de la compañía. Este ejemplo se visualiza en la Figura 27:

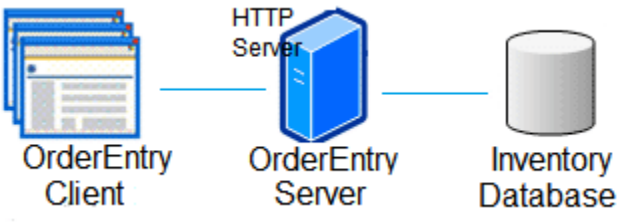

Figura 27: Ejemplo de utilización Servlets

La especificación original de *Servlets* fue creada por Sun Microsystems (la versión 1.0 fue terminada en junio de 1997). A partir de la versión 2.3, la especificación de *Servlet* fue desarrollada siguiendo el *Java Community Process*.

El API *Servlet* recoge el conjunto de clases e interfaces que implementan la interfaz entre el servidor y los *servlets*. El paquete *javax.servlet* se compone de clases genéricas, válidas para cualquier protocolo. El paquete *javax.servlet.http* extiende la API y la concreta para el protocolo HTTP. Estas APIs son comunes para todos los servidores capaces de ejecutar *servlets*. *[17], [18]*

2.6.1.1 Características principales

Las características principales de los *Servlets* son:

- Se ejecutan en una máquina virtual dentro del proceso del servidor.
- Gestionados por un “motor de *servlets*”.
- Cada petición HTTP recibida se procesa en un hilo, e invoca un método del *servlet*, utilizando Servlet API
- Además de servidores de HTTP, pueden extender cualquier tipo de servidor (por ejemplo, FTP).

2.6.1.2 Ciclo de vida

El ciclo de vida de un *Servle*t se divide en los siguientes puntos:

- El cliente solicita una petición a un servidor vía URL.
- El servidor recibe la petición.

  - Si es la primera, se utiliza el motor de *Servlets* para cargarlo y se llama al método *init()*.
  - Si ya está iniciado, cualquier petición se convierte en un nuevo hilo. Un *Servlet* puede manejar múltiples peticiones de clientes.
- Se llama al método *service()* para procesar la petición devolviendo el resultado al cliente.
- Cuando se apaga el motor de un *Servlet* se llama al método *destroy(),* que lo destruye y libera los recursos abiertos.

2.6.1.3 Ventajas y desventajas

Las ventajas del uso de esta tecnología se recogen en la siguiente Tabla 6:

Potencia: - APIs de Java: acceso a red, hilos, manipulación de imágenes, acceso a bases de datos, compresión de datos, internacionalización, RMI, CORBA, serialización de objetos, acceso a directorio, etc. - Utilización de código externo, incluido EJBs.	**Eficiencia:** - Instancia permanentemente cargada en memoria por cada *servlet*. - Ejecución de peticiones mediante invocación de un método. - Cada petición se ejecuta en un hilo. - Mantiene automáticamente su estado y recursos externos: conexiones a bases de datos, conexiones de red, etc.
Seguridad: - Lenguaje java: máquina virtual, chequeo de tipos, gestión de memoria, excepciones, etc. - Gestor de seguridad de Java.	**Extensibilidad y flexibilidad:** - API Servlet extensible. - Filtros (cadenas de servlets). - Integrable con JSP (*Java Server Pages*).
Elegancia: - Código java: modular, orientado a objetos, limpio y simple. - API *servlets*: potente y fácil de utilizar.	**Integración:** - Integración fuerte entre *servlets* y servidor: permite colaboración entre ambos.
Portabilidad entre plataformas y servidores.	Comunidad grande de desarrolladores.

Tabla 6: Ventajas de la utilización de Servlets

Las desventajas del uso de esta tecnología se recogen en la siguiente Tabla 7:

Concurrencia: Por ser un único objeto ejecutado desde varios hilos, es necesario gestionar la concurrencia.	Código del programa mezclado con código HTML del documento:
Cualquier cambio requiere recompilar el *servlet.*	Bastante limitado a lenguaje Java

Tabla 7: Desventajas de la utilización de Servlets

Un *servlet* no resulta adecuado para presentación (escribir directamente la salida HTML) porque el código HTML está entremezclado dentro del código Java. Por ejemplo:

```
PrintWriter out = response.getWriter();
out.println("<html>");
out.println("<head>");
```

Esto es incómodo para cambiar el código HTML y no facilita la división de tareas entre diseñadores de HTML y programadores. La alternativa consiste en escribir directamente el código HTML con poco código Java incrustado, utilizando JSPs.

2.6.2 JavaServer Pages (JSP)

JavaServer Pages (JSP) es una tecnología Java que permite generar contenido dinámico para web, en forma de documentos HTML, XML o de otro tipo.

Esta tecnología es un desarrollo de la compañía Sun Microsystems. La especificación JSP 1.2 fue la primera que se liberó y en la actualidad está disponible la especificación JSP 2.1.

Una página JSP es una página (X)HTML que incorpora ciertos elementos dinámicos: etiquetas especiales y pequeños fragmentos de código. El código HTML aparece a la salida sin modificaciones, y los elementos dinámicos se evalúan o ejecutan en el servidor en el momento de construcción de la respuesta. *[19]*

2.6.2.1 Características principales

Las características principales de los JSPs son:

- Tiene la misma portabilidad, eficiencia y potencia que *Servlets*.

- Elegancia: separación del código del documento del código de la aplicación (*JavaBeans*).
- Facilidad de programación: gestiona automáticamente sesiones, atributos de peticiones y respuestas, redirecciones, ciclo de vida de *JavaBeans*, etc.
- Integrable con *Servlets*.

2.6.2.2 Java Bean

Un *Java Bean* es una clase Java que cumple el siguiente convenio:

- Contiene propiedades (normalmente atributos de instancia privados).
- El acceso a las propiedades se realiza mediante métodos de acceso *get*, *set* e *is*.
- Tiene siempre un constructor sin argumentos (aunque podría tener más constructores).

2.6.3 Arquitectura de una aplicación con *Servlets* y JSP

Se considera que una aplicación Web realiza tareas de procesado y presentación. Los *Servlets* son adecuados para procesado, mientras que las páginas JSP son adecuadas presentación. *[20]*

Una aplicación Web puede combinar *Servlets* y páginas JSP, de la siguiente forma:

- Procesado de parámetros de la petición: *Servlets.*
- Acceso a bases de datos: *Servlets*.
- Lógica de la aplicación: *Servlets*.
- Presentación (vistas): JSP.

El modelo de funcionamiento es el siguiente:

1. El cliente envía la petición HTTP a un *servlet*.
2. El *servlet* procesa la petición. Si es necesario, se conecta a la base de datos.
3. El *servlet* redirige la petición a un JSP. Si es necesario, añade *beans* como parámetros.
4. El JSP lee los parámetros y devuelve la respuesta formateada visualmente al usuario.

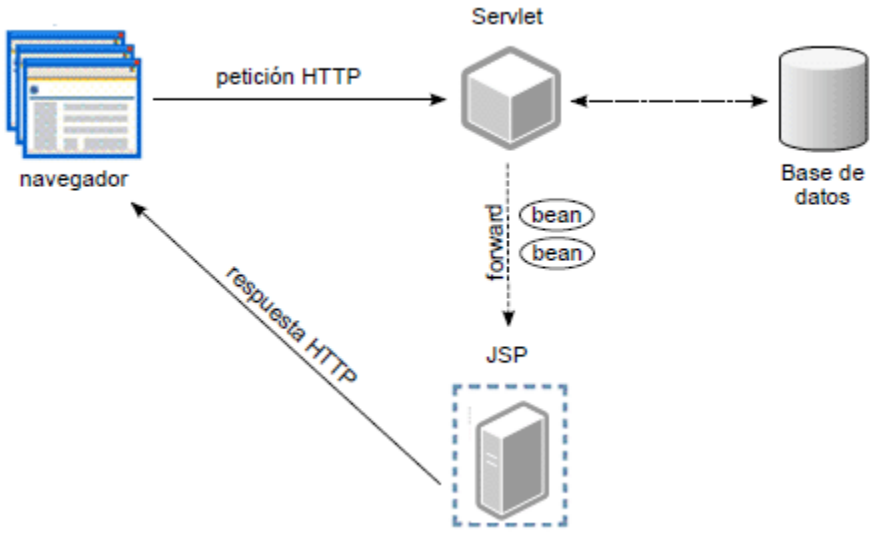

Figura 28: Modelo de funcionamiento de una aplicación web con Servlets y JSP

2.6.4 Apache Tomcat

El servidor Apache Tomcat es una implementación de referencia de las tecnologías *Java Servlet* y *JavaServer Pages* (JSP).

Es mantenido y desarrollado por miembros de la *Apache Software Foundation* y voluntarios independientes. Los usuarios disponen de libre acceso a su código fuente y a su forma binaria en los términos establecidos en la *Apache Software Licence.*

Tomcat es un servidor web con soporte de *servlets* y JSPs. El motor de *servlets* de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Figura 29: Logo de Apache Tomcat

Tomcat puede funcionar como servidor web por sí mismo. Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual Java.

Tomcat empezó siendo una implementación de la especificación de los *servlets* comenzada por James Duncan Davidson, que esperaba que el proyecto se convirtiese en software de código abierto. Dado que la mayoría de los proyectos de este tipo tienen libros de *O'Reilly* asociados con un animal en la portada, quiso ponerle al proyecto nombre de animal. Eligió *Tomcat* (gato), pretendiendo representar la capacidad de cuidarse por sí mismo, de ser independiente. *[21]*

2.6.5 JavaDB Apache Derby

Apache Derby es un sistema gestor de base de datos relacional escrito en Java que puede ser empotrado en aplicaciones Java y utilizado para procesos de transacciones *online*, es decir cliente-servidor. Apache Derby es un proyecto de código abierto licenciado bajo la Apache 2.0 *License,* que actualmente se distribuye como Sun Java DB. *[22]*

Figura 30: Logo de Apache Derby

Sus características son:

- APIs para JDBC y SQL.
- Su código mide alrededor de 2MB comprimido.
- Soporta cifrado completo, roles y permisos. Además posee SQL SCHEMAS para separar la información en una única base de datos y control completo de usuarios.
- Soporta internamente *procedures,* cifrado y compresión.
- Trae soporte multilenguaje y localizaciones específicas.
- Transacciones y recuperación ante errores.
- Posee tres productos asociados a la marca:
    - *Derby Embedded Database Engine*: El motor propiamente dicho.
    - *Derby Network Server*: Permite convertir Derby en una base de datos que sigue el modelo cliente-servidor tradicional.
    - *Database Utilities*: Un paquete de utilidades.

2.7 Teléfono móvil Nokia 6131 NFC

La realización de este proyecto se ha desarrollado utilizando teléfonos móviles en los que poder descargar las aplicaciones con las distintas funcionalidades desarrolladas.

El modelo de teléfono móvil utilizado es el Nokia 6131 NFC. *[23]*

Figura 31: Imagen del teléfono Nokia 6131 NFC

Lo más importante a destacar de este móvil es la tecnología NFC integrada:

- Soporta el API JSR-257 para las aplicaciones que utilicen funciones NFC.
- Soporta tanto lectura y escritura en los formatos más populares de etiquetas: Mifare 1K, 4K, Ultralight, Topaz, Jewel, FeliCa.
- Permite aplicaciones *peer-to-peer*.
- Contiene un chip seguro integrado (*Global Platform* 2.1.1), que permite ser tratado como una tarjeta ISO 14443 Tipo A o Mifare 4K.

En el siguiente apartado, se detallan las características completas de este móvil.

2.7.1 Características del Nokia 6131 NFC

Tamaño: - Peso: 104 g - Dimensiones: 92 x 47 x 20 mm	**Pantalla e interfaz de usuario:** - Pantalla principal: 2.2” QVGA TFT - Pantalla cubierta externa: 1.36” TFT
Imagen: - Cámara de 1.3 megapíxeles con zoom digital 8x	**Mensajería:** - SMS y MMS - MMS 1.2

- Visor de pantalla completa para la pantalla principal y la pantalla exterior	- Envío de emails con adjuntos - Mensajería instantánea - "*Push to talk* " (Pulse para hablar) - Mensajería de audio Nokia Xpress: envío de mensajes de voz y clips de audio vía MMS
Multimedia: - Reproductor de música con soporte de múltiples formatos - Radio estéreo FM - Vídeo streaming para formato 3GPP - Tonos de llamada en vídeo. - Tonos de llamada MIDI	**Funciones de memoria:** - 11 MB memoria de usuario - Capacidad de memoria expandible con tarjeta de memoria microSD de 2GB
Aplicaciones: - Java MIDP 2.0 juegos y aplicaciones	**Conectividad:** - *Near field communication* (NFC) con capacidades de lectura, escritura y *peer-to-peer* - Chip de seguridad integrado (Global Platform 2.1.1) - Nokia PC Suite con USB, Bluetooth, y conectividad IrDa - Bluetooth version 2.0 con la especificación *Enhanced Data Rate* - Ranura para tarjeta de memoria microSD - Conector *Pop-Port*™ con conectividad USB
Buscador: - Navegador XHTML integrado - Descarga inteligente de contenidos, OMA	**Transferencia de datos:** - EDGE (EGPRS): multislot class 10 - GPRS: multislot class 10
***Personal Information Management* (PIM):** - Calendario configurable, con posibilidad de ver notas personales	**Características NFC:** - Capacidad de pago sin contacto y de *ticketing*. - Acceso a servicios móviles e

en el modo de espera activa - Listas personales, notas, calculadora, alarma y cronómetro	información con un simple toque - Uso de la especificación Java requerida (JSR 257) para aplicaciones NFC de terceros
Características de voz: - Servicio *Push to talk* - Marcación por voz con SIND (altavoz independiente con nombre de marcación) - Altavoz manos libres integrado de alta calidad - Comandos de voz - Grabación de datos	**Servicios digitales:** - Temas de la interfaz de usuario como fondos de pantalla o salvapantallas animados, y tonos de llamada - Tonos de llamada: vídeos, tonos MP3, MIDI, alertas y tonos de juegos - Posibilidad de descarga a través de OTA de toda la multimedia descrita

Tabla 8: Características del teléfono móvil Nokia 6131 NFC

2.7.2 Nokia NFC SDK

El SDK Nokia 6131 NFC permite a los desarrolladores crear, emular y ejecutar aplicaciones Java (MIDlets) para el teléfono móvil Nokia 6131 NFC.

El SDK incluye el *Contactless Communication API* (API de Comunicación Sin Contacto, JSR-257), que habilita el uso de las características NFC de este móvil. Además del estándar JSR-257 API, incluye extensiones adicionales para varias tecnologías de etiquetas y conexión *peer to peer.*

El desarrollador puede usar un IDE para crear compilar y empaquetar *MIDlets*. El comportamiento de estos *MIDlets* se puede comprobar a través del emulador, directamente desde el IDE, o también puede ser usado por sí solo como un programa aparte, usando el "lanzador de *MIDlet*" o directamente desde la línea de comandos.
Este emulador además proporciona simulación de etiquetas NFC y de tarjetas inteligentes Java. También pueden ser utilizados lectores de tarjetas externos.

Para facilitar su uso, viene acompañado de: APIs de Java, MIDlets de ejemplo y documentación.

Capítulo 3 : Implementación de utilidades necesarias

En este capítulo se realiza una descripción técnica sobre las implementaciones realizadas en este proyecto. En concreto, sobre las implementaciones de utilidades NFC y de utilidades Bluetooth.

3.1 Implementación de utilidades NFC

En este apartado se explica de manera detenida la implementación de la comunicación NFC presente en este proyecto.

Las implementaciones NFC que se han desarrollado en los teléfonos móviles utilizados, modelo Nokia 6131 NFC, son de dos tipos:

- Comunicación pasiva entre el móvil del cliente y una tarjeta NFC.

- Comunicación activa entre el móvil del cliente y el otro móvil que funciona como lector NFC.

Figura 32: Comunicación pasiva entre el móvil del cliente y una tarjeta NFC

Figura 33: Comunicación activa entre el móvil del cliente y el otro móvil que funciona como lector NFC

El desarrollo de utilidades NFC en los dispositivos móviles, se ha realizado utilizando la plataforma J2ME. En el apartado 2.4 del presente documento se detallan las características de esta plataforma. J2ME es la edición de Java enfocada a dispositivos electrónicos con capacidades computacionales y gráficas muy reducidas, tales como teléfonos móviles.

Para el desarrollo de las aplicaciones móviles realizadas en J2ME con capacidades NFC, se ha utilizado el API de Comunicaciones Sin Contacto (*Contactless Communication API*), JSR 257, que se detalla a continuación:

3.1.1 *Contactless Communication API* - JSR 257

En este apartado se detalla el API JSR 257 desarrollado por Java para las comunicaciones NFC. Primero se describe qué es un JSR, y a continuación se detalla más en profundidad el API.

3.1.1.1 *Java Specification Request*, JSR

El Proceso de la Comunidad Java, o *Java Community Process*, establecido en 1998, es un proceso formalizado el cual permite a las partes interesadas a

involucrarse en la definición de futuras versiones y características de la plataforma Java.

El proceso JCP conlleva el uso de *Java Specification Request* (JSR), documentos formales que describen las especificaciones y tecnologías propuestas para que sean añadidas a la plataforma Java.

Un JSR final suministra una implementación de referencia la cual da una implementación libre de la tecnología en código fuente y un Kit de Compatibilidad de Tecnología para verificar la especificación de la API.

3.1.1.2 Introducción al JSR 257

El desarrollo de las aplicaciones móviles en las que interviene la tecnología NFC de este proyecto, se han llevado a cabo utilizando el lenguaje de programación J2ME con el API adicional JSR 257, utilizado para comunicaciones sin contacto, basadas en proximidad. *[24]*

A continuación se describe la estructura del API JSR 257. Las clases e interfaces se dividen en cinco paquetes:

- *javax.microedition.contactless:* Provee descubrimiento de *targets* (objetivos con los que comunicarse*)* y clases comunes a todos ellos.

- *javax.microedition.contactless.ndef:* Contiene clases e interfaces necesarias para la comunicación con etiquetas que tienen datos en formato NDEF.

- *javax.microedition.contactless.rf:* Permite comunicación con etiquetas RFID que no tienen datos en formato NDEF.

- *javax.microedition.contactless.sc:* Habilita la comunicación con *smartcards* (tarjetas inteligentes), tanto con la interna que incluye el móvil como con tarjetas externas compatibles con el estándar ISO 14443, basándose en los comandos APDU.

- *javax.microedition.contactless.visual:* Proporciona formas de leer la información almacenada en códigos de barras (etiquetas visuales) y de generar dichas etiquetas. Se soportan los tipos más comunes de códigos de barras y matriciales como el *DataMatrix* o el *QR Code*

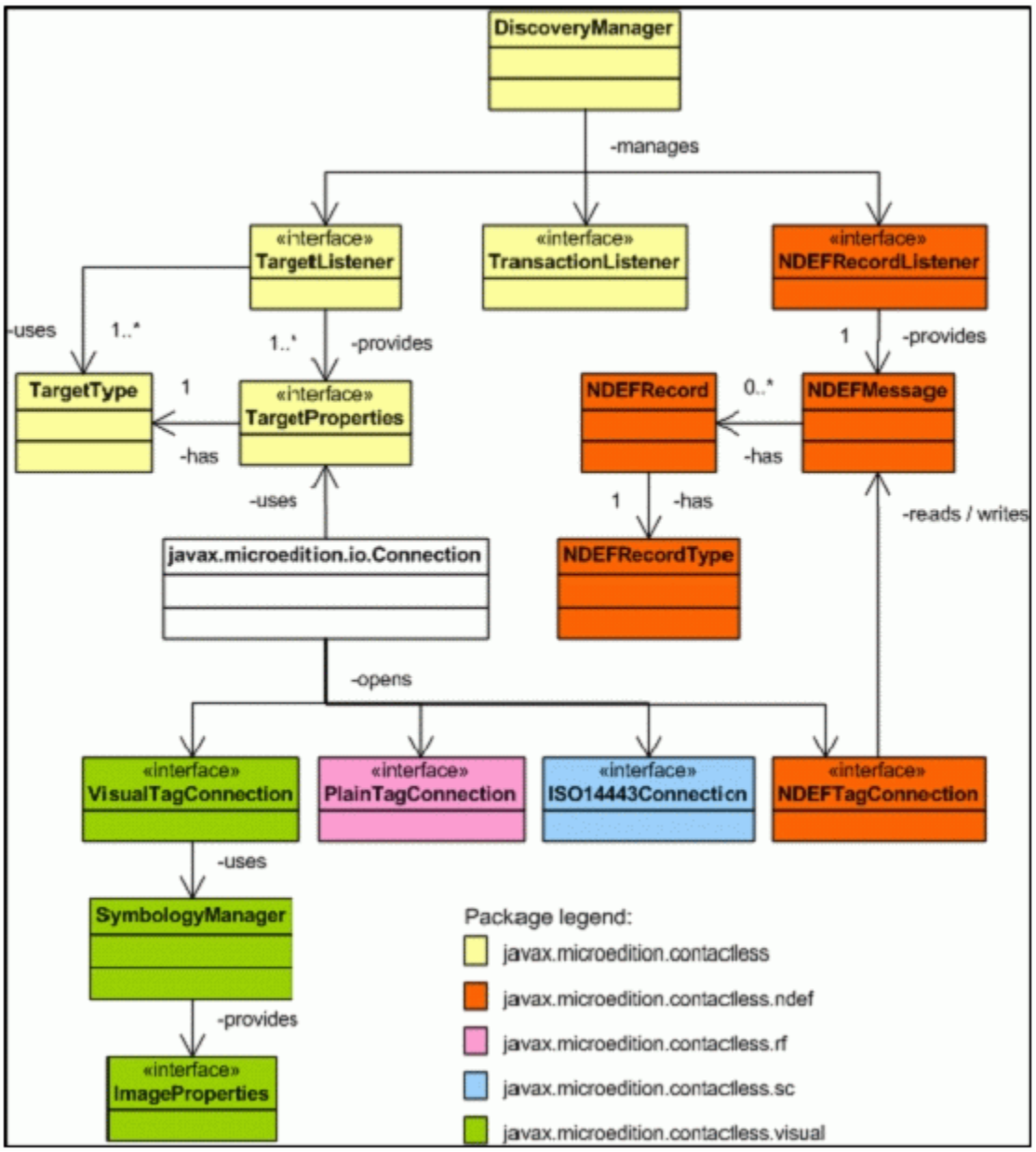

Figura 34: Arquitectura de paquetes del Contactless Communicacion API

En este proyecto, se han utilizado los dos primeros paquetes: *javax.microedition.contactless y javax.microedition.contactless.ndef,* que se detallan más ampliamente a continuación:

3.1.1.3 Paquete javax.microedition.contactless

El descubrimiento de *contactless targets (*dispositivos *"*objetivo" sin contacto con los que establecer una comunicación*)* es el punto de partida de este API. Una vez se descubierto, una aplicación se puede comunicar con él.

En la Figura 34 se muestras las clases e interfaces de este paquete, que son:

- Clase *DiscoveryManager*: Ofrece mecanismos para el descubrimiento de dispositivos “objetivos sin contacto” y maneja los diferentes tipos de *listeners* (escuchadores de eventos) para este API.

- Clase *TargetType*: Recoge todos los tipos de “objetivos sin contacto” soportados por este API como: *ISO14443_CARD, NDEF_TAG, RFID_TAG, VISUAL_TAG*.

- Interfaz *TagConnection*: Toda conexión que pueda ser usada para la comunicación con etiquetas RFID o tarjetas inteligentes externas, debe extender de esta interfaz.

- Interfaz *TargetListener*: Provee mecanismos para que la aplicación sea notificada cuando el hardware (teléfono móvil) descubre “objetivos sin contacto”.

- Intefaz TargetProperties: Recoge las propiedades que son comunes a todos los “objetivos sin contacto”, soportados por esta especificación.

- Interfaz *TransactionListener*: Notifica a la aplicación cuando hay actividad del elemento seguro, cuando el móvil está en modo de emulación de tarjetas.

3.1.1.4 Paquete javax.microedition.contactless.ndef

El NFC Forum especifica un formato de empaquetado de datos para el intercambio de información entre dispositivos y etiquetas NFC. Este formato se basa en registros NDEF, utilizado por este API para la conexión a “objetivos sin contacto” sin la necesidad de saber el tipo físico de los mismos.

En la Figura 34 se muestras las clases e interfaces de este paquete, que son:

- Clase *NDEFMessage*: Representa un mensaje NDEF que puede contener uno o más registros NDEF. Esta clase permite la manipulación de los registros NDEF en memoria.

- Clase *NDEFRecord*: Consiste de un tipo, formato de tipo e identificador para la cabecera. Es la abstracción de un registro NDEF definido por el NFC Forum.

- Interfaz *NDEFRecordListener*: Provee mecanismos de notificación de descubrimiento de registros NDEF.

- Clase *NDEFRecordType*: Encapsula los tipos de un registro NDEF. Almacena el nombre y formato de los nombres del registro. Sigue las reglas de los RTD's definidos por el NFC Forum.

- Interfaz *NDEFTagConnection*: Define la funcionalidad básica para el intercambio de información formateada según el NFC Forum. Provee mecanismos para escribir y leer datos sin conocer el tipo físico del "objetivo sin contacto".

3.1.1.5 La comunicación *peer-to-peer* dentro de este API

La fecha de la primera versión de este API es el 17 Octubre de 2006. La versión 1.1 de mantenimiento es de 10 Julio del 2009. En estas dos versiones no se incluye la comunicación *peer-to-peer*.

Sí que se hace referencia a ella en el Apéndice D: "Ítems futuros" de los documentos de las versiones citadas.

En este apéndice se menciona que durante el desarrollo de estas especificaciones algunos ítems originalmente planeados a ser incluidos, finalmente no formaron parte de esta especificación.

Esta especificación provee un API de Java para las características definidas por el NFC Forum. Algunas de esas características no fueron completamente finalizadas (es decir, aún no han sido estandarizadas) por el NFC Forum durante el desarrollo de este API, y por ello no se incluyeron. Una de estas características es la comunicación NFC *peer-to-peer*.

La comunicación *half-duplex* NFC Forum *peer-to-peer*, permite intercambiar datos entre dos dispositivos NFC.

Para la realización de este proyecto, en el que es necesario este tipo de comunicación, se ha utilizado la extensión de este API facilitada por Nokia dentro de su SDK, que permite la comunicación *P2P* entre dos dispositivos Nokia.

3.1.2 Utilización del Contactless Communication API.

En este apartado vamos a ver los pasos básicos para la utilización de este API.

En el desarrollo de este proyecto, se utiliza este API para obtener los datos bancarios de un cliente a través de nuestro teléfono móvil.

Figura 35: Detalle de móvil acercándose a una etiqueta NFC

Es decir, se acerca una tarjeta NFC externa que contiene información bancaria de un cliente a nuestro móvil, y en ese momento se realiza una comunicación NFC pasiva de tal forma que el móvil lee los datos de la tarjeta que se le ha acercado.

Esta funcionalidad está implementada en el teléfono móvil de nuestro cliente. Para el uso del API, lo primero que se ha de hacer es importar los paquetes que se van a utilizar. En nuestro caso, son los siguientes:

```
import javax.microedition.contactless.*;
import javax.microedition.contactless.ndef.*;
```

Además, la clase principal del cliente donde se va a incluir la funcionalidad citada anteriormente, ha de implementar la interfaz *TargetListener* y *NDEFRecordListener*, que habilitan la posibilidad de que la aplicación sea notificada cuando se descubre una tarjeta NFC y un registro NDEF, respectivamente:

```
public class ClienteMIDlet extends MIDlet implements CommandListener,
Runnable, TargetListener,  NDEFRecordListener {
```

En el siguiente fragmento de código, se realizan estas dos operaciones:

- Registro al descubrimiento de dispositivos. Selecciona con qué tipos de destinos se va a comunicar. En este caso, el tipo seleccionado es NDEF_TAG.
- Registro al descubrimiento de un tipo determinado registro NDEF: Para que la aplicación sea notificada al detectar un destino determinado, se registra un escuchador de eventos para registros NDEF del tipo que se desee. Los tipos pueden ser: estándar del NFC Forum, texto, URI o *Smart Poster*, tipo

MIME, URI, tipo definido por el usuario según las normas de nombres establecidas, tipo desconocido o tipo que representa a un registro vacío. En este caso, se trata de MIME *"text/plain".*

```
private DiscoveryManager dm;
  dm = DiscoveryManager.getInstance();
        try {
            // Registramos la etiqueta NDEF al TargetListener
            dm.addTargetListener(this, TargetType.NDEF_TAG);
            // Y configuramos al NDEFRecordListener
            dm.addNDEFRecordListener(this, new NDEFRecordType(
                    NDEFRecordType.MIME, "text/plain"));
        } catch (IllegalStateException e) {
            formtag.append(e.toString());
        } catch (ContactlessException e) {
            formtag.append(e.toString());
        }
```

Método *targetDetected()*

El siguiente fragmento corresponde al método *targetDetected()*. Este es el método que recibirá la notificación una vez se haya hecho la detección. Se recibe como parámetro un *array* porque se pueden haber detectado varios destinos del tipo elegido al mismo tiempo.

Cada objeto *TargetProperties* contiene la URL a través de la cual abrir la conexión, así como el identificador del destino. Las conexiones se abren utilizando la *Generic Connection Framework*, de la misma forma que las conexiones HTTP o Bluetooth.

A continuación se crea un *NDEFRecord* para ser añadido al *NDEFMessage*, del tipo MIME *"text/plain*", a partir de una cadena creada en el propio código como ejemplo de datos contenidos en la tarjeta.

Después se escribe el mensaje en la tarjeta NFC que se acerca al móvil, que sobrescribirá todos los datos contenidos.

```
/** Metodo que notifica cuando una etiqueta NDEF es detectada */
    public void targetDetected(TargetProperties[] properties) {
            NDEFTagConnection conn = null;
            try {
                // Se Abre conexión con la etiqueta NDEF
                conn = (NDEFTagConnection)
Connector.open(properties[0].getUrl());

                String config0 =
"Pedro#Perez#667894563#4b#visa#euro6000#servired";
                String[] tags = new String[1];

                tags[0] = config0;
```

```
// Se crea un NDEF Record para ser añadido al NDEF Message
 NDEFRecord recordToWrite = new NDEFRecord(new NDEFRecordType(
           NDEFRecordType.MIME, "text/plain"), null, tags[0].getBytes());

// Se crea un NDEF Message y se añade el record creado.
  NDEFMessage write_container = new NDEFMessage();
  write_container.appendRecord(recordToWrite);

// Escribimos el NDEF Message en la etiqueta
conn.writeNDEF(write_container);

  formtag.append("Datos escritos: " + tags[0]);

            } catch (IOException e) {
                ...
            }
    }
```

Metodo *recordDetected()*

El siguiente método es *recordDetected()*, que es activado por *NDEFRecordListener*, cuando se descubren registros NDEF del tipo seleccionado.

Se recibirá el mensaje NDEF completo, no solo el registro que ha activado el escuchador de eventos. La aplicación puede procesarlo y realizar las operaciones que quiera con los datos. Este caso es más sencillo que el caso anterior, ya que no es necesario establecer una conexión, pero por este mismo motivo, tampoco es posible realizar escrituras.

```
    /**
     * Método que notifica un NDEF message detectado
     */
    public void recordDetected(NDEFMessage mes) {
            // Se recoge la información del NDEF message
            NDEFRecord rec[] = mes.getRecord(new NDEFRecordType(
                    NDEFRecordType.MIME, "text/plain"));

            byte[] p = rec[0].getPayload();
            String cadena = new String(p);
            formtag.append("Información guardada: " + cadena);
            ...
        }
```

3.1.3 *Peer-to-peer Communication:* extensión JSR 257 API

Como se comentó en el apartado 3.1.1.5, la comunicación *peer-to-peer* no está contemplada en el JSR 257, ya que el NFC Forum todavía no ha estandarizado un protocolo P2P. Sin embargo, Nokia lo incorpora como una extensión al API.

El Nokia 6131 NFC SDK, descrito en el apartado 2.7.2, soporta una serie de extensiones del API. A partir de ellas, se pueden crear y compilar *MIDlets* que usen estas extensiones, pero no es posible emularlos a través del simulador de Nokia, es necesario probarlo a través de dispositivos Nokia 6131 NFC reales.

La extensión usada que nos permite la comunicación *peer to peer* se encuentra en el siguiente paquete: *com.nokia.nfc.p2p.* Este paquete proporciona un interfaz para comunicar dispositivos a través de NFCIP. *[25]*

3.1.4 Utilización de la comunicación *peer-to-peer*

En este apartado se muestran los pasos básicos para la comunicación *peer to peer* entre dos dispositivos activos.

NFCIP permite a dos dispositivos comunicarse entre ellos cuando se encuentran cerca. La comunicación sigue un protocolo simple de pregunta-respuesta. La interfaz *NFCIPConnection* está localizada en el paquete *com.nokia.nfc.p2p.*

En el desarrollo de este proyecto, se utiliza esta capacidad para la comunicación entre los dos dispositivos móviles que intervienen en el sistema: un teléfono móvil del cliente, y un teléfono móvil que funciona como lector.

Figura 36: Ilustración de la comunicación NFCIP entre los dos dispositivos móviles

Es decir, el protocolo de comunicación que se establece entre estos dos elementos del sistema, que sigue una comunicación de este tipo, les permite intercambiar, mediante este mecanismo de “enviar-recibir”, toda la información pertinente para poder realizar un pago, que es el propósito final de este proyecto. Para que se establezca esta comunicación es necesario acercar un dispositivo al otro.

Por tanto, esta funcionalidad está implementada en forma de aplicación *MIDle*t en el teléfono móvil de nuestro cliente y en el teléfono móvil lector.

Un *MIDlet* puede funcionar siendo *initiator* (iniciador) o *target* (objetivo):

- Un *initiator* establece la conexión, envía datos y recibe una respuesta.
- Un *target* establece una conexión, recibe datos, y envía una respuesta.

Cuando se establece la conexión, la URL especifica qué *MIDle*t será el *initiator* y qué *MIDlet* será el *target.* Se debe de usar una URL predefinida:

```
String INITIATOR_URL = "nfc:rf;type=nfcip;mode=initiator";
String TARGET_URL = "nfc:rf;type=nfcip;mode=target";
```

La conexión se abre utilizando *Connector*. El método *open()* bloqueará el hilo hasta que se encuentre el otro extremo de la comunicación.

En este caso no puede utilizarse el *DiscoveryManager/TargetListener* para que se notifique a la aplicación cuando se detecte otro dispositivo NFC. La conexión se abre de la siguiente manera:

```
NFCIPConnection conn = (NFCIPConnection) Connector.open(url);
```

Uno de los dispositivos, actuará como *initiator*: El *MIDlet* enviará datos y recibirá una respuesta.

```
NFCIPConnection conn = (NFCIPConnection)
Connector.open("nfc:rf;type=nfcip;mode=initiator");
byte[] message = ...
conn.send(message);
byte[] response = conn.receive();
```

El otro dispositivo, actuará como *target.* El *MIDlet* recibirá datos y enviará una respuesta.

```
NFCIPConnection conn = (NFCIPConnection)
      Connector.open(“nfc:rf;type=nfcip;mode=target");
byte[] message = conn.receive();
byte[] response = ...;
conn.send(response);
```

Al final, la conexión deberá ser cerrada:

```
conn.close();
```

Pueden ser realizadas tantas secuencias de pregunta-respuesta como sea necesario. Por cada llamada a *send()* (enviar), debe de haber en el otro extremo, una llamada a *receive()* (recibir).

3.2 Implementación de utilidades Bluetooth

En este apartado se explicará de manera detenida la implementación de la comunicación Bluetooth presente en este proyecto. Esta comunicación se realiza entre un dispositivo móvil y un ordenador, tal y como se muestra en la siguiente Figura 37.

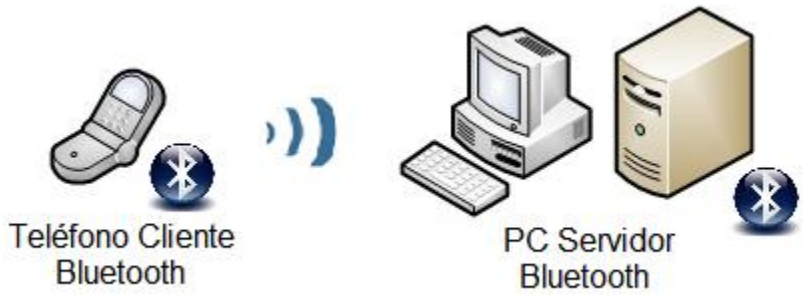

Figura 37: Comunicación Bluetooth en este proyecto

Para el desarrollo de la aplicación móvil, realizada en J2ME, con capacidad de Cliente Bluetooth, se ha utilizado *Java APIs for Bluetooth*, JSR 82.

Para el desarrollo de la aplicación con capacidad de servidor Bluetooth en el PC, se ha utilizado la librería *Bluecove*. Este desarrollo se detalla a continuación:

3.2.1 *Java APIs for Bluetooth* - JSR 82

Java APIs for Bluetooth es una especificación Java ME que permite a *MIDlets* Java (aplicaciones para móviles Java), usar Bluetooth en sus dispositivos.

La especificación fue desarrollada bajo el *Java Community Process* como JSR 82. *[26]*

Este API está dividido en dos paquetes totalmente independientes:

- *javax.bluetooth*: define clases e interfaces básicas para el descubrimiento de dispositivos, descubrimiento de servicios, conexión y comunicación. La comunicación a través de *javax.bluetooth* es a bajo nivel, mediante flujos de datos o mediante la transmisión de cadenas de bytes.

- *javax.obex:* permite manejar el protocolo de alto nivel OBEX (*OBject EXchange)*. Este protocolo es muy similar a HTTP y es utilizado sobre todo para el intercambio de archivos.

La plataforma principal de desarrollo del API JSR-82 es J2ME. El API ha sido diseñado para depender de la configuración CLDC. Sin embargo existen implementaciones para poder hacer uso de este API en J2SE.

En una comunicación Bluetooth existe un dispositivo que ofrece un servicio (servidor) y otros dispositivos acceden a él (clientes). Este API permite realizar todas las acciones que intervienen en una comunicación Bluetooth:

Un cliente Bluetooth deberá realizar las siguientes acciones:

- Búsqueda de dispositivos: La aplicación realizará una búsqueda de los dispositivos Bluetooth a su alcance que estén en modo conectable.
- Búsqueda de servicios: La aplicación realizará una búsqueda de servicios por cada dispositivo.
- Establecimiento de la conexión: Una vez encontrado un dispositivo que ofrece el servicio deseado se conecta a él.
- Comunicación: Ya establecida la conexión, se podrá leer y escribir en ella.

Por otro lado, un servidor Bluetooth deberá hacer las siguientes operaciones:

- Crear una conexión servidora.
- Especificar los atributos de servicio.
- Abrir las conexiones clientes.

3.2.2 *Bluecove* API

Bluecove es una librería de Java para Bluetooth basada en la implementación JSR-82. Puede ser usada en la *Java Standard Edition* (J2SE) 1.1 o superiores, y está habilitada para varios sistemas operativos.*[27]*

Proporciona una interfaz JSR-82 Java para los siguientes perfiles Bluetooth:

- SDAP - *Service Discovery Application Profile*
- RFCOMM - *Serial Cable Emulation Protocol*
- L2CAP - *Logical Link Control and Adaptation Protocol*
- GOEP - *Generic Object Exchange (OBEX) Profile*

Ya que es una implementación de la librería JSR-82, proporciona el mismo API usado para las aplicaciones en J2ME, que es para lo que específicamente está definido.

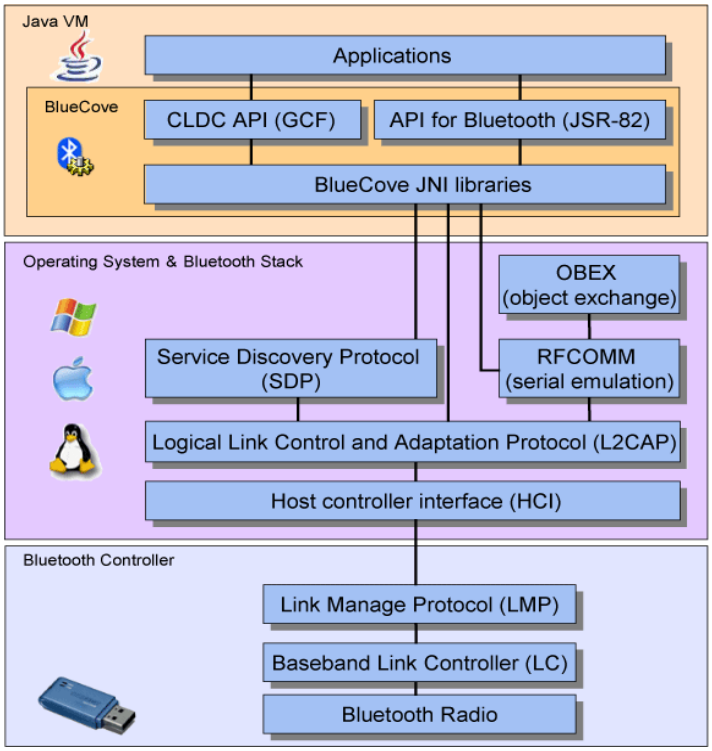

Figura 38: Librería *Bluecove* en la torre de protocolos Bluetooth

3.2.3 Utilización de *Java APIs for Bluetooth* - JSR 82

En este proyecto, esta comunicación se ha llevado a cabo entre uno de los dispositivos móviles Nokia 6131 NFC y el PC.

El dispositivo móvil Nokia 6131 NFC, por tanto, a través de J2ME, utilizará el API JSR-82, funcionando como Cliente. El PC, a través de J2SE, utilizará la implementación del API JSR-82 desarrollada por *Bluecove*, funcionando como Servidor.

El perfil Bluetooth a utilizar es el *Serial Port Profile* (perfil del puerto serie): SPP.

Un perfil Bluetooth, como se detalló en el apartado 2.2 del presente documento, es una descripción de un comportamiento general que los dispositivos pueden utilizar para comunicarse, formalizados para favorecer un uso unificado. La forma de utilizar las capacidades de Bluetooth se basa, por tanto, en los perfiles que soporta cada dispositivo.

El perfil puerto serie se basa en la especificación 07.10 de ETSI por medio del protocolo RFCOMM.

El protocolo RFCOMM proporciona una emulación del Puerto Serie RS-232 entre dos dispositivos Bluetooth, estableciéndose una sesión.

La aplicación del PC, en este caso, ofrecerá un servicio basado en el perfil de puerto serie, SPP, actuando como servidor. La aplicación *MIDlet* del dispositivo móvil iniciará una conexión al servicio SPP, actuando como cliente. Entonces se establecerá una comunicación Bluetooth entre ambos dispositivos.

A continuación se muestran los pasos básicos para la comunicación Bluetooth entre los dos dispositivos.

3.2.4 Cliente Bluetooth

Primero se verá el lado del cliente, el lado del teléfono móvil. Como se describió anteriormente, un cliente Bluetooth deberá realizar las siguientes operaciones:

Inicialmente, deberá realizar una búsqueda de los dispositivos que estén a su alcance y, a continuación, una búsqueda de servicios por cada dispositivo encontrado. Después de encontrar el servicio del dispositivo deseado, establece una conexión con él y, por tanto, a partir de ese momento, puede comenzar una comunicación e intercambio de datos.

Todo este proceso, que se resume visualmente en la siguiente Figura 39, a continuación se detalla paso por paso.

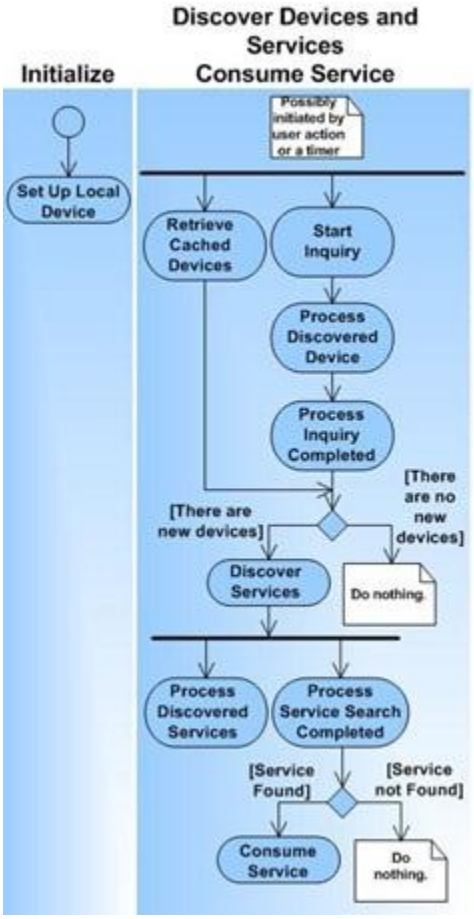

Figura 39: Proceso de búsqueda de dispositivo y servicio en un cliente Bluetooth

3.2.4.1 Búsqueda de dispositivos

En el metodo *startApp()* de nuestro *MIDlet* principal tenemos la llamada al método *startDeviceInquiry().*

Para realizar la búsqueda de dispositivos cercanos, se debe de obtener un objeto *DiscoveryAgen*t que será único y se obtiene a través del objeto *LocalDevice*, que representa al dispositivo local*. DiscoveryAgent* permite realizar y cancelar búsquedas de dispositivos y de servicios, y obtener listas de dispositivos ya conocidos:

Método *retrieveDevices()*

A través del método *retrieveDevices(),* se obtiene un *array* de dispositivos remotos ya conocidos o encontrados en búsquedas anteriores.

Metodo *startInquiry()*

A través del método *startInquiry(),* se comienza una nueva búsqueda de dispositivos. Este método requiere dos argumentos:

- El primer argumento es un entero que especifica el modo de conectividad que deben tener los dispositivos a buscar. Este valor deberá ser:
  - *DiscoveryAgent.GIAC (General/Unlimited Inquiry Access Code):* Conectividad ilimitada
  - *DiscoveryAgent.LIAC* (*Limited Dedicated Inquiry Access Code) :* Conectividad limitada.

- El segundo argumento es un objeto que implemente *DiscoveryListener*. A través de éste serán notificados los dispositivos que se vayan descubriendo. En nuestro caso se refiera a la clase principal (*this*), ya que implementa a la interfaz "*DiscoveryListener*"

```
private void startDeviceInquiry() {
            try {

                  DiscoveryAgent agent =
                  LocalDevice.getLocalDevice().getDiscoveryAgent();
                  agent.startInquiry(DiscoveryAgent.GIAC, this);
            } catch (Exception e) {
                  …
            }
}
```

La interfaz, *DiscoveryListener*, se usa por tanto, para que el dispositivo notifique eventos a la aplicación cada vez que se descubre un dispositivo, un servicio, o se finaliza una búsqueda. Esta interfaz tiene los siguientes métodos:

- *public void deviceDiscovered(RemoteDevice rd, DeviceClass c)*
- *public void inquiryCompleted(int c)*
- *public void servicesDiscovered(int transID, ServiceRecord[] sr)*
- *public void serviceSearchCompleted(int transID, int respCode)*

Los dos últimos métodos serán llamados durante un proceso de búsqueda de servicios, que se verá más adelante. Vemos más en detalle los dos primeros métodos, que serán llamados durante el proceso de búsqueda de dispositivos:

Método *deviceDiscovered()*

Cada vez que se descubre un dispositivo se llama a este método. El primer argumento es un objeto de la clase *RemoteDevice* que representa el dispositivo encontrado. El segundo argumento nos permitirá determinar el tipo de dispositivo encontrado. Como se ve en el siguiente fragmento, dentro del método se añade el objeto *RemoteDevice* a una lista para que se muestre a continuación.

```
 public void deviceDiscovered(RemoteDevice btDevice, DeviceClass cod) {
                deviceList.addElement(btDevice);
 }
```

Método *inquiryCompleted()*

Este método es llamado cuando la búsqueda de dispositivos ha finalizado. Nos pasa un argumento entero indicando el motivo de la finalización. Este argumento podrá tomar los valores:

- *DiscoveryListener.INQUIRY_COMPLETED,* si la búsqueda ha concluido con normalidad.
- *DiscoveryListener.INQUIRY_ERROR*, si se ha producido un error en el proceso de búsqueda.
- *DiscoveryListener.INQUIRY_TERMINATED,* si la búsqueda fue cancelada.

```
  public void inquiryCompleted(int discType) {
        anterior.append("Búsqueda completada. Por favor, seleccione un
dispositivo al que conectarse.");
        makeDeviceSelectionGUI();
    }
```

3.2.4.2 Búsqueda de servicios

Dentro del método *inquiryCompleted(),* mostrado en el fragmento de código anterior, se llama al método *makeDeviceSelectionGUI().* El propósito de esta llamada es que, una vez la búsqueda haya sido completada, se seleccione un dispositivo, a partir de la lista mostrada, con el que establecer una búsqueda de servicio:

```
private void makeDeviceSelectionGUI() {
      final List devices = new List("Seleccione un dispositivo:",
      List.IMPLICIT);
      for (int i = 0; i < deviceList.size(); i++) {
         devices.append(
             getDeviceStr((RemoteDevice) deviceList.elementAt(i)), null);
      }

      devices.setCommandListener(new CommandListener() {

        public void commandAction(Command arg0, Displayable arg1) {

            startServiceSearch((RemoteDevice)
            deviceList.elementAt(devices.getSelectedIndex()));
        }
      });
      display.setCurrent(devices);
```

```
 }
```

Método searchServices()

Para comenzar la búsqueda usaremos *searchServices()* de la clase *DiscoveryAgent*, que es llamado dentro del método *startServiceSearch()*:

```
protected UUID uuid = new UUID(0x0110); //Puerto Serie

private void startServiceSearch(RemoteDevice device) {
        try {
            UUID uuids[] = new UUID[]{uuid};
            getAgent().searchServices(null, uuids, device, this);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

Los argumentos del método *searchServices(int[] attrSet, UUID[] uuidSet, RemoteDevice btDev, DiscoveryListener discListener), son:*

- Primer argumento: *array* de enteros con el que se especifica los atributos de servicio en los que se está interesado.
- Segundo argumento: *array* de identificadores de servicio. Permite especificar los servicios en los que se está interesado. En este caso, el Puerto Serie.
- Tercer argumento: dispositivo remoto sobre el que se va a realizar la búsqueda.
- Último argumento: *this,* objeto que implementa *DiscoveryListener* que será usado para notificar los eventos de búsqueda de servicios.

El método *searchServices()* devolverá un entero que identificará la búsqueda. Este valor entero sirve para saber a qué búsqueda pertenecen los eventos *servicesDiscovered()* y *serviceSearchCompleted(),* de la interfaz *DiscoveryListener*

Método *serviceSearchCompleted()*

El *método serviceSearchCompleted(int transID, int respCode)*, es llamado cuando se finaliza un proceso de búsqueda. El primer argumento identifica el proceso de búsqueda. El segundo argumento indica el motivo de finalización de la búsqueda, que pueden ser:

Método *servicesDiscovered()*

Este método notifica que se han encontrado servicios. El primer argumento es un entero que identifica el proceso de búsqueda. El segundo argumento es un

array de objetos *ServiceRecord*. Un objeto *ServiceRecord* describe las características de un servicio Bluetooth.

```
    public void servicesDiscovered(int transId, ServiceRecord[] records)
{
        for (int i = 0; i < records.length; i++) {
            ServiceRecord rec = records[i];
            String url = rec.getConnectionURL(connectionOptions, false);
            handleConnection(url);
        }
    }
```

3.2.4.3 Establecimiento de la conexión

El API *javax.bluetooth* permite usar dos mecanismos de conexión: SPP y L2CAP. Mediante SPP se obtiene un *InputStream* y un *OutputStream*. Mediante L2CAP se envían y reciben *arrays* de bytes. Para abrir la conexión, se hace uso de la clase *javax. microedition.io.Connector.*

En este caso, se quiere establecer una conexión Puerto Serie: SPP.

Método *getConnectionURL()*

Dentro del método *servicesDiscovered()* mostrado en el anterior fragmento de código, se obtiene la URL necesaria para realizar la conexión a través del método *getConnectionURL()* de un objeto *ServiceRecord*, es decir, del servicio encontrado deseado.

```
int connectionOptions = ServiceRecord.NOAUTHENTICATE_NOENCRYPT;

ServiceRecord rec = records[i];
String url = rec.getConnectionURL(connectionOptions, false);
handleConnection(url);
```

Este método requiere dos argumentos:

- El primero indica si se debe autenticar y/o cifrar la conexión.
- El segundo argumento es un booleano que especifica si nuestro dispositivo debe hacer de maestro (*true*) o bien no importa si es maestro o esclavo (*false*).

Con la URL necesaria, llamamos al método *handleConnection()* en donde podremos realizamos la conexión necesaria, a través de:

```
conn = (StreamConnection) Connector.open(url);
```

3.2.4.4 Comunicación Bluetooth

El método *Connector.open(url)* devuelve un objeto distinto según el tipo de protocolo usado. En el caso de un cliente SPP devolverá un *StreamConnection*.
A partir del *StreamConnection* se obtienen los flujos de entrada y de salida, con los que ya se puede establecer una comunicación. En el siguiente fragmento de código se muestra el método *handleConnection()* anteriormente citado, donde se establece la comunicación.

```
private void handleConnection(final String url) {

        try {
            conn = (StreamConnection) Connector.open(url);

            out = conn.openDataOutputStream();
            in = conn.openDataInputStream();

            readString();
            writeString("Hello");

} catch (Throwable e) {
           …
        }
}

public void writeString(String str) throws IOException {
        out.writeUTF(str);
        out.flush();
}
public String readString() throws IOException {
        String t;
        t = in.readUTF();
        return t;
}
```

3.2.5 Servidor Bluetooth

El papel de Servidor Bluetooth, lo realizará el otro extremo de la comunicación Bluetooth: el PC.

Como se explicó anteriormente, un servidor Bluetooth deberá hacer las siguientes operaciones: Crear una conexión servidora, especificar los atributos de servicio y abrir las conexiones clientes

En el siguiente fragmento de código se muestra la implementación de estos tres pasos, siendo éste un desarrollo mucho más sencillo que el realizado para el dispositivo cliente.

El primer paso para ofrecer un servicio a través de Bluetooth, es poner el dispositivo en modo visible. Esto se hace a través de la clase *LocalDevice*:

```
LocalDevice device = LocalDevice.getLocalDevice();
device.setDiscoverable(DiscoveryAgent.GIAC);
```

Para crear una conexión servidora se pasa la URL *"localhost"* al método *Connector.open().* De este modo la URL deberá comenzar por *"btspp://localhost:"*

Además del host de la URL, se debe indicar el UUID que identifica el servicio. Posteriormente se indica el nombre del servicio y otros parámetros, como por ejemplo el requerimiento o no de autenticación.

```
String url = "btspp://localhost:" + UUID + ";name=PCServerCOMM";
```

El método *Connector.open()* devuelve un "notificador", que en el caso de SPP será un objeto *StreamConnectionNotifier:*

```
StreamConnectionNotifier notifier = (StreamConnectionNotifier)
                          Connector.open(url);
serverLoop(notifier);
```

Llegados a este punto ya se puede escuchar conexiones clientes a través de *acceptAndOpen()* del siguiente modo:

```
private void serverLoop(StreamConnectionNotifier notifier) {
   try {
            handleConnection(notifier.acceptAndOpen());
   } catch (Exception e) {
           […]
    }
}
```

Con el objeto *StreamConnection,* ya se puede establecer una comunicación del mismo modo que en las aplicaciones cliente.

```
private void handleConnection(StreamConnection conn) {

      out = conn.openDataOutputStream();
      in = conn.openDataInputStream();

      try {
          writeString("PC -> Reader: Hello");
          readString();

            […]
}
```

Capítulo 4 : Descripción funcional del sistema

En este capítulo se realiza una introducción al esquema de funcionamiento general realizado en este proyecto.

4.1 Introducción

En este proyecto, se ha diseñado y desarrollado un sistema simulado de la realización de un pago a través de un dispositivo móvil con la tecnología NFC.

Para el desarrollo del sistema, como se puede observar en la Figura 40, se ha realizado el siguiente esquema general, dividiéndose en cuatro bloques: Cliente, Tienda, Red Bancaria y Banco.

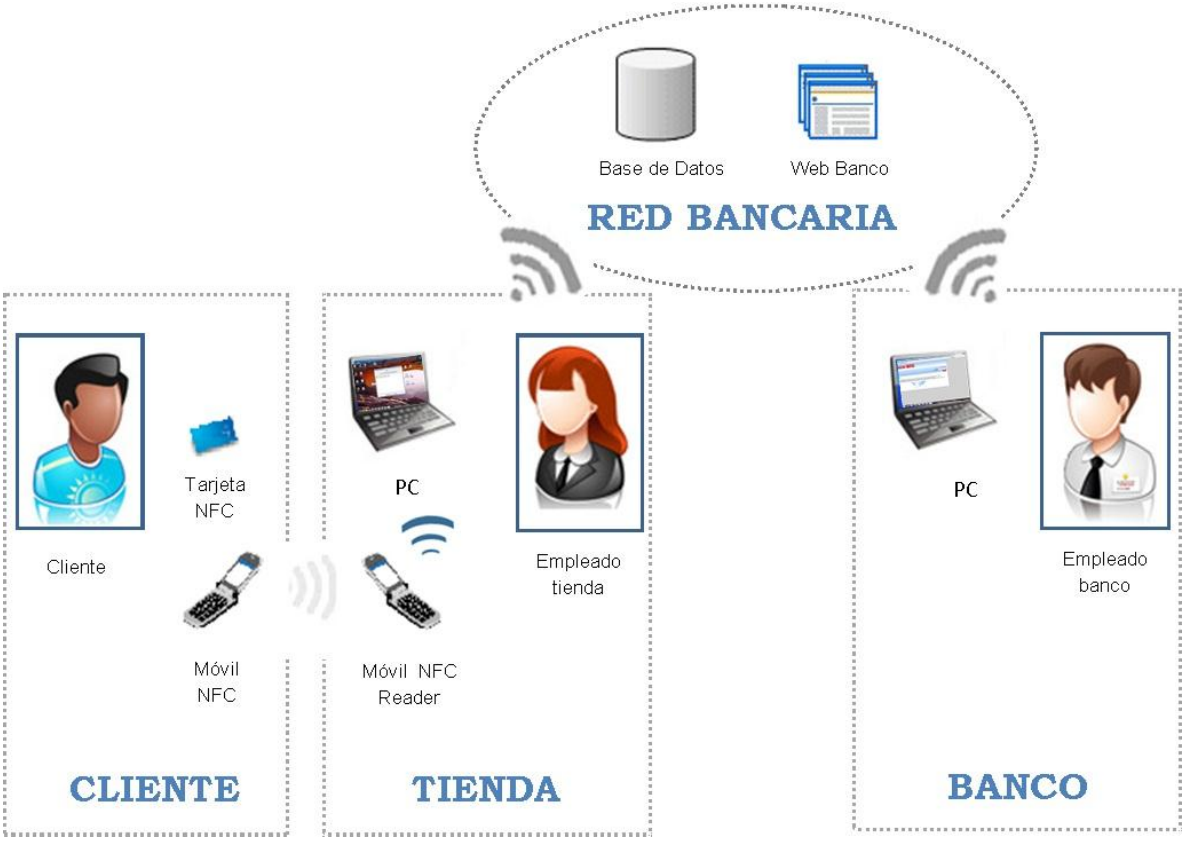

Figura 40: Diagrama general del sistema

- Bloque Cliente:

  - Usuario: Cliente.
  - Elementos que utiliza el usuario:
    - Teléfono Nokia 6131 NFC.
    - Tarjeta sin contacto NFC.
  - Aplicaciones:
    - Aplicación MIDlet “Cliente” contenida en el teléfono móvil.

- Bloque Tienda:

  - Usuario: Empleado tienda.
  - Elementos que utiliza el usuario:
    - Teléfono Nokia 6131 NFC.
    - Ordenador.
  - Aplicaciones:
    - Aplicación MIDlet “Reader” contenida en el teléfono móvil.
    - Aplicación Java Swing “PCServer” contenida en el ordenador.

- Bloque Red Bancaria: Se compone de:

    - Servidor que aloja una aplicación web del Banco “Red Bancaria”.
    - Base de datos bancaria “redBancaria”.

- Bloque Banco:

    - Usuario: Empleado banco.
    - Elementos que utiliza el usuario:
        - Ordenador.
    - Aplicaciones: Accede a una aplicación web que reside en el servidor de la red bancaria.

La descripción general del funcionamiento es la siguiente:

Un cliente con su teléfono móvil NFC, para realizar el pago en una tienda, se dirige al mostrador, donde será atendido por el empleado de esta tienda.

El móvil del cliente contiene sus datos bancarios (tarjetas bancarias). A través de él podrá realizar un pago, con el simple acto de acercarlo a un lector NFC.

En el mostrador estará el dispositivo lector NFC, encargado de establecer una comunicación NFCIP con el móvil NFC Cliente, y una comunicación Bluetooth con el ordenador de la tienda, para poder efectuar el pago. Además, para que sea efectivo, el ordenador de la tienda se conectará a la base de datos de la red bancaria donde éste será gestionado.

A la vez, se tendrá una aplicación web, donde un empleado de la entidad bancaria podrá realizar gestiones de las tarjetas bancarias registradas por los clientes.

Este funcionamiento se verá detallado a continuación.

4.2 Funcionalidades de cada bloque y aplicación

En este apartado se presenta una explicación a la funcionalidad de cada bloque y de cada aplicación de cada bloque para poder entender y tener una visión general del sistema propuesto en este proyecto.

4.2.1 Bloque Cliente.

Este bloque representa la figura del Cliente, que utiliza un teléfono móvil NFC Nokia para realizar un pago. Para ello, el cliente tendrá que realizar el simple acto de acercar su móvil a un dispositivo lector NFC.

4.2.1.1 Aplicación Cliente

Dentro de este teléfono se ejecuta una aplicación MIDlet llamada “Cliente”, que tiene las siguientes funcionalidades:

4.2.1.1.1 Manejo de los datos bancarios del cliente

Como se describió en el apartado 2.1.3.7, el denominado Elemento Seguro, es un chip con características de seguridad similares a las encontradas en las tarjetas inteligentes y que se encarga de procesar de forma segura las transacciones. Puede encontrarse en la electrónica del móvil, en la tarjeta SIM, o en una tarjeta externa.

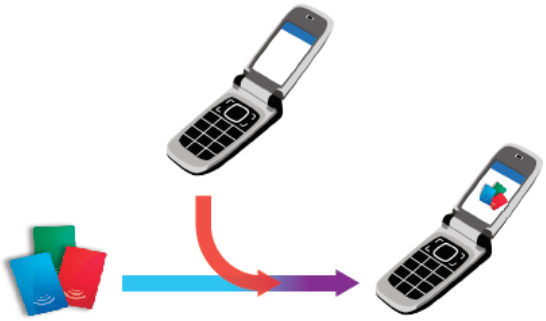

Figura 41: Ilustración del concepto “Tarjeta inteligente sin contacto + Teléfono móvil”

En este proyecto, se utilizaría el elemento seguro para guardar la información bancaria del cliente, la misma que se guardaría en una tarjeta bancaria.

El problema en este caso, ha sido que no se disponía ni del hardware ni de los permisos necesarios en el dispositivo móvil Nokia, para poder grabar o acceder a datos que se alojan en el Elemento Seguro del móvil, que se encuentra en la electrónica del mismo.

En este entorno simulado propuesto, se ha desarrollado como solución a ello, dos tipos de “modos de funcionamiento”:

- El modo de funcionamiento “Elemento Seguro”, que recoge los datos bancarios del cliente de la propia aplicación instalada en el Cliente.

- El modo de funcionamiento “Simulación”, que recoge los datos bancarios del cliente de una tarjeta NFC externa poniéndose en contacto con el móvil, es decir, funcionando el móvil como lector de tarjetas NFC, y estableciéndose una comunicación NFC pasiva entre la tarjeta y el móvil.
  Es motivo de la existencia de este modo que el cliente disponga de estos dos elementos en el escenario general.
  Esta comunicación hace uso del *Contactless Communication API JSR-257*, que se describe en el apartado 3.1.1, y se detalla su implementación en el apartado 3.1.2

Figura 42: Elementos necesarios en el modo de funcionamiento “Simulación”, donde se recogen los datos bancarios de una tarjeta NFC pasiva externa

4.2.1.1.2 Comunicación con el lector NFC

La comunicación con el Lector NFC es la otra parte lógica de la aplicación Cliente. Se trata de una comunicación NFCIP, también llamada *peer-to-peer*, que hace uso de la extensión del API JSR-257, que se describe en el apartado 3.1.3. La utilización de este API para establecer esta comunicación, viene detallada en el apartado 3.1.4

Esta comunicación se establece acercando un dispositivo con el otro. Su propósito, como se explicó anteriormente, es establecer un protocolo para formalizar un pago entre ambos elementos. La aplicación cliente obtiene la información del cliente, de cualquier de las dos formas de funcionamiento descritas en el apartado anterior, y a través de un protocolo de comunicación, se intercambia datos con el elemento lector NFC, que se encuentra del lado de la Tienda, manejado por el usuario “Empleado tienda”.

El elemento lector NFC, viene representado por otro teléfono móvil Nokia 6131. Esta decisión se explica en el siguiente apartado, que detalla el escenario de la Tienda.

Figura 43: Esquema de los dos dispositivos Nokia que intervienen en la comunicación NFCIP

El protocolo que usan ambos elementos viene descrito en el apartado 5.5

4.2.2 Bloque Tienda

Este bloque representa la parte de la Tienda que, gestionada por el usuario "Empleado tienda", dispone de los elementos necesarios para cobrar a sus clientes aplicando la tecnología NFC.

En un escenario real, el pago con el teléfono móvil NFC, se puede realizar de dos formas:

- A través de un datáfono NFC (NFC + lógica de pago)
- A través de un Lector NFC, comunicado con un ordenador (que implementa la lógica de pago)

Ejemplos de estos dos escenarios reales de pago se pueden observar a continuación:

Figura 44: Ilustración del pago a través de un datafono

Figura 45: Ilustración del pago a través de un Lector NFC

Como para la realización de este proyecto no se ha dispuesto de un datáfono ni de un Lector NFC, esto se simulará mediante otro móvil NFC que realizará la funcionalidad de comunicación NFCIP, más un ordenador que simulará la lógica del propio datáfono, la lógica necesaria para realizar el pago. El móvil NFC Lector y el PC están interconectados mediante conexión *Bluetooth.*

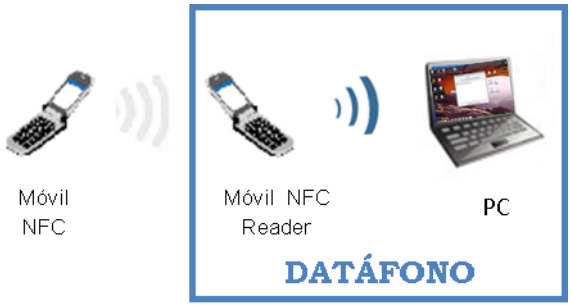

Figura 46: Esquema que muestra a los elementos que, integrados, funcionan como un datáfono

En este bloque existen dos aplicaciones:

- La aplicación “Reader”, contenida en el móvil NFC Reader, que se encarga de la comunicación NFCIP con el móvil NFC Cliente y la comunicación Bluetooth con el PC.

- La aplicación “PCServer” contenida en el ordenador de la tienda, que se encarga de representar la lógica del datafono, y está conectada con la red Bancaria y con el móvil NFC Reader.

4.2.2.1 Aplicación Reader

Esta aplicación se encuentra en el móvil NFC Reader, es decir, el móvil NFC del bloque de la tienda.

Esta aplicación permanece a la escucha, y establece una comunicación con el Móvil NFC Cliente cuando éste se le acerca.

La aplicación Reader también establece una comunicación con la aplicación PCServer, a través de Bluetooth. Esta aplicación PCServer está conectada a la base de datos de la red bancaria, con el objetivo de formalizar el pago.

4.2.2.2 Aplicación PCServer

La aplicación PCServer contenida en el ordenador de la tienda, se encarga de representar la lógica del datáfono.

Si un extremo del protocolo de comunicación a través del cual se realiza el pago es el móvil NFC Cliente, el otro extremo es esta aplicación, ya que se encarga de procesar la información enviada por el cliente (leída a través del móvil lector y reenviada por éste a través de Bluetooth al pc), procesarla, y contestarle.
Para procesar esta información, y realizar la lógica con la que se efectúa el pago, requiere más datos:

- El importe de la compra, que es introducido por el empleado de la tienda en la aplicación. A este empleado también se le da el derecho, dado que estamos en un entorno simulado, de exigir el modo de funcionamiento con el que quiere funcionar al teléfono móvil cliente (Elemento Seguro, o Simulación). Esta idea de modo de funcionamiento fue presentada en el apartado 4.2.1.1.1

- Los datos bancarios del cliente en el momento actual. Esta aplicación está conectada a una base de datos a través de la cual se realizan las transacciones de pago, y las gestiones necesarias para que la operación de pago pueda realizarse satisfactoriamente.

Es decir, la aplicación PCServer no sólo establece comunicación con el móvil NFC Reader a través de comunicación Bluetooth, sino que por otro lado tiene acceso a una base de datos del Banco, en la que poder leer, actualizar tarjetas y datos de las tarjetas de los clientes.

El protocolo de comunicación entre ambos extremos, como se referenció anteriormente, se encuentra en el apartado 5.5.

4.2.2.3 Pago *on-line* y pago *off-line*

En el diseño de este protocolo, se ha querido simular ambos tipos de pago: *online* y *offline.*

El pago *offline* se trata de un pago con tarjeta en el que propio pago y transacción bancaria no se hace en realidad operativo en ese momento, sino que se hace más tarde. Esto está pensado para no crear colas de pago y agilizar el proceso.

La utilización de pagos *offline* para ciertos servicios como autopistas o las máquinas de billetes de metro, se determina por la Asociación Nacional de Medios de Pago. Esta asociación es la que realiza un acuerdo por encima de la entidad bancaria que presta servicio, para tomar este tipo de decisiones. Está formada por entidades bancarias.

El pago *offline* u *online* dependerá de un precio mínimo. Esta decisión no está gestionada por la Asociación Nacional de medios de pago, sino que se trata de una decisión del cliente que realiza ventas. Las razones por las que se quiera hacer, pueden ser para que las compras de cantidades inferiores a un límite no ralenticen la cola de pago, y se prefieran hacer *offline*, corriendo el riesgo de que esta tarjeta no sea válida.

El precio mínimo estará acordado entre el cliente vendedor y el banco que le presta servicio para realizar los pagos con tarjeta, es decir, el propietario del datáfono.

En este sistema, se ha fijado un precio mínimo de 8 euros. Se diferencia, además, un pago *offline* u *online* en si es requerido el PIN de la tarjeta para realizar la compra.

4.2.3 Bloque Red Bancaria

Este bloque representa una red de una entidad bancaria a través de la cual podemos acceder a los datos bancarios de los clientes, que se encuentran compilados en una base de datos. A través de la aplicación web RedBancaria podremos realizar consultas y gestiones de las tarjetas de los clientes.

4.2.3.1 Base de datos redBancaria

En esta base de datos se encuentran los datos de los clientes de nuestro sistema simulado de pago.

El modelo de la base de datos se ilustra en la siguiente **Figura 47: Estructura de la base de datos**Figura 47:

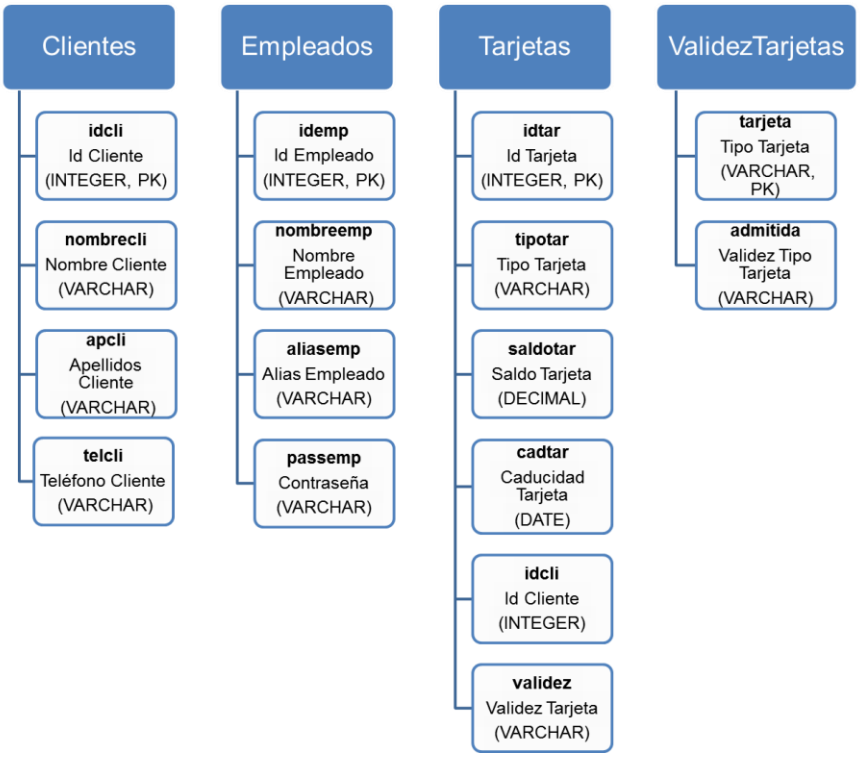

Figura 47: Estructura de la base de datos

La base de datos ha sido diseñada tal que:

- Un Cliente es representado con los siguientes datos: un número de identificación, su nombre, sus apellidos y su número de teléfono móvil.

- Un Empleado del banco es representado con los siguientes datos: un número de identificación, su nombre, su 'alias' o nombre de usuario y su contraseña de empleado.

- Una tarjeta de un cliente es representada con: un número de identificación, el tipo de tarjeta (mastercard, 4b, etc), el saldo actual de la tarjeta, la caducidad de esa tarjeta, el número de identificación del cliente a la que pertenece, y la validez de esta tarjeta actualmente.

Por último se tiene una tabla llamada "ValidezTarjetas" con la que se simula los tipos de tarjetas que la red bancaria soporta.

4.2.3.2 Aplicación web RedBancaria

La aplicación web RedBancaria reside en un servidor de la red bancaria. Su función principal es ser una interfaz para un empleado del banco, para la gestión de las tarjetas de los clientes.

A través de ella, por tanto, se pueden realizar actualizaciones de los datos de las tarjetas, es decir, se podrá actualizar y gestionar la base de datos redBancaria.

4.2.4 Bloque Banco

Este bloque representa la figura del Empleado del banco. El empleado del banco accede a través de su ordenador en su puesto de trabajo a la aplicación web RedBancaria, que reside en un servidor de la propia entidad bancaria. A través de esa aplicación, podrá realizar las gestiones pertinentes para mantener actualizados los datos de las tarjetas de los clientes.

Capítulo 5 : Detalles de implementación del sistema

En este capítulo se detalla de una forma más profunda los distintos elementos que forman parte del sistema. En concreto, se presenta para cada aplicación desarrollada, su diagrama de componentes, su diagrama funcional de clases, y un breve manual de uso.

5.1 Aplicación móvil Cliente

5.1.1 Diagrama de componentes

El diagrama de la Figura 48 muestra el paquete de la aplicación Cliente, que se ejecuta en el teléfono móvil NFC del cliente. Esta aplicación se compone de cuatro clases Java:

- ClienteMIDlet: el *MIDlet* principal, donde se gestiona el funcionamiento principal, la lectura pasiva NFC y la interfaz gráfica de la aplicación
- CConexion: Clase que sirve para gestionar la comunicación NFCIP
- ElementoSeguro: Clase que simula un objeto del modo Elemento Seguro
- Simulacion: Clase que simula el objeto del modo Simulación (tarjeta externa)

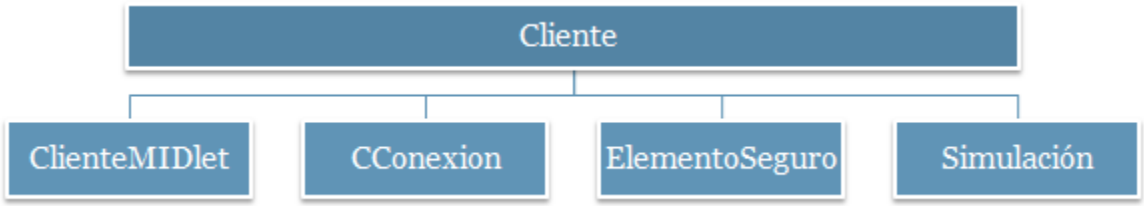

Figura 48: Diagrama de componentes de la aplicación Cliente

5.1.2 Diagrama funcional de clases

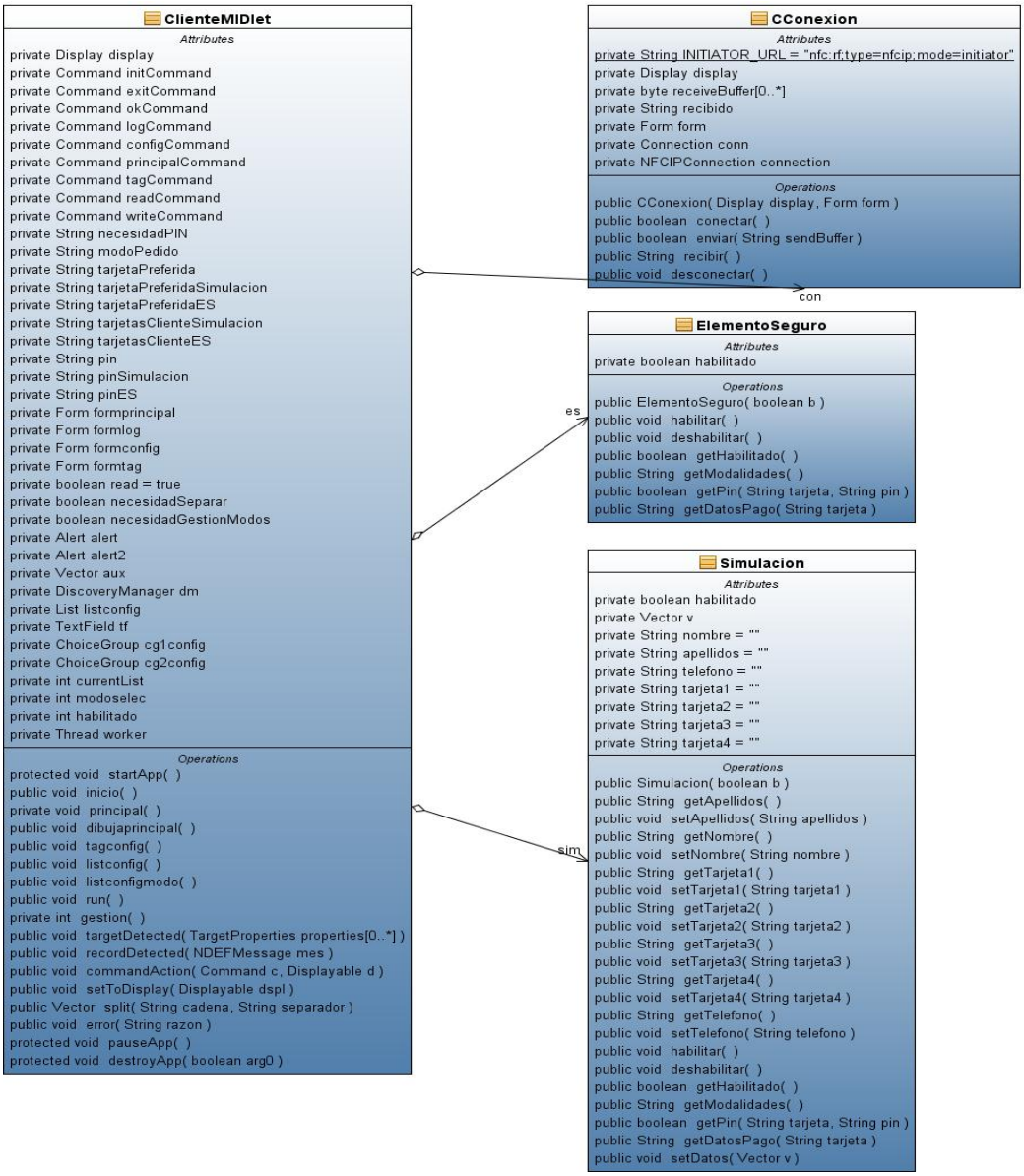

Figura 49: Diagrama funcional de la aplicación Cliente

En este diagrama, se puede observar que en la clase ClienteMIDlet se realiza la funcionalidad principal. Desde esta clase se implementa un objeto de la clase CConexion, para poder realizar la comunicación NFCIP, y un objeto de las clases Simulación y Elemento seguro, que funcionan como interfaz de los datos bancarios del cliente.

5.1.3 Manual de usuario

A continuación se describe cómo debe manejar la aplicación móvil cliente el usuario Cliente.

La situación inicial sería un usuario que desea realizar un pago en el escenario propuesto en este proyecto. Para ello, inicia a través de su móvil la aplicación “Cliente”.

En el menú principal, aparece la siguiente información:

- Se debe pulsar “Configuración” para configurar los modos de pago
- Se debe pulsar “Iniciar” para realizar el pago.

Además, se muestra la configuración de modos de funcionamiento actual: qué modos están habilitados, y si está configurada alguna tarjeta como preferida en cada uno de ellos.

En el apartado introductorio a esta aplicación, 4.2.1.1, se presentó el concepto diseñado en este proyecto de “Modo de funcionamiento”, que representa la forma de obtener los datos del cliente: desde la propia aplicación instalada en el teléfono cliente (modo “Elemento Seguro”), o a través de una etiqueta NFC externa poniéndola en contacto con nuestro móvil (modo “Simulación”).

En el móvil, por tanto, se puede habilitar o deshabilitar la funcionalidad de estos modos. En el caso de querer trabajar en el Modo Simulación, antes de entrar en su menú de configuración se debe de poner en contacto con el móvil la tarjeta NFC externa con los datos bancarios del cliente, para que aparezcan en dicho menú.

Al realizar la configuración de los modos de pago, además de su habilitación, también se puede elegir la tarjeta preferida o predefinida a utilizar, es decir, la tarjeta con la cual se quiere realizar el pago, debiéndose de introducir el PIN de ésta para poder elegirla.

En la siguiente Figura 50, se muestra los pasos a seguir para la configuración de una tarjeta de un modo de funcionamiento, en este caso, de la tarjeta euro6000 del modo Elemento Seguro: En el menú principal, se selecciona la opción de

"Configuración". Dentro del menú de configuración, como vemos en el detalle de la figura, se debe de elegir el Modo en el que realizar la configuración pertinente.

El siguiente menú muestra los campos a rellenar necesarios para poder cambiar la configuración de este modo: si queremos que esté habilitado o no, y en caso afirmativo, si se quiere, se puede elegir la tarjeta que queremos marcar como predefinida. Para poder elegirla se debe de introducir su PIN.

Si el proceso es correcto, aparecerá en el menú principal la configuración elegida (en este ejemplo, se muestra en la última pantalla el estado final de la configuración: el modo 'Elemento Seguro' ha sido habilitado con la tarjeta 'euro6000').

Figura 50: Uso de la aplicación móvil Cliente I

Si desde el menú principal se elige “Opciones”, se podrá acceder a los menús “Leer Tarjeta” o “Iniciar”, como se puede observar en la siguiente Figura 51.

El menú “Leer Tarjeta”, muestra los datos leídos, si se ha acercado una etiqueta o tarjeta NFC externa al móvil.

Al menú “Iniciar” se debe de acceder cuando se haya realizado todas las configuraciones y se quiera realizar la operación de pago.

Para ello, cuando se accede al menú Iniciar, se debe de acercar el móvil al aparato lector o datáfono (que, en este caso, es el móvil Lector de la tienda), que estará a la escucha, para poder establecer la comunicación NFCIP entre ambos para realizar el pago.

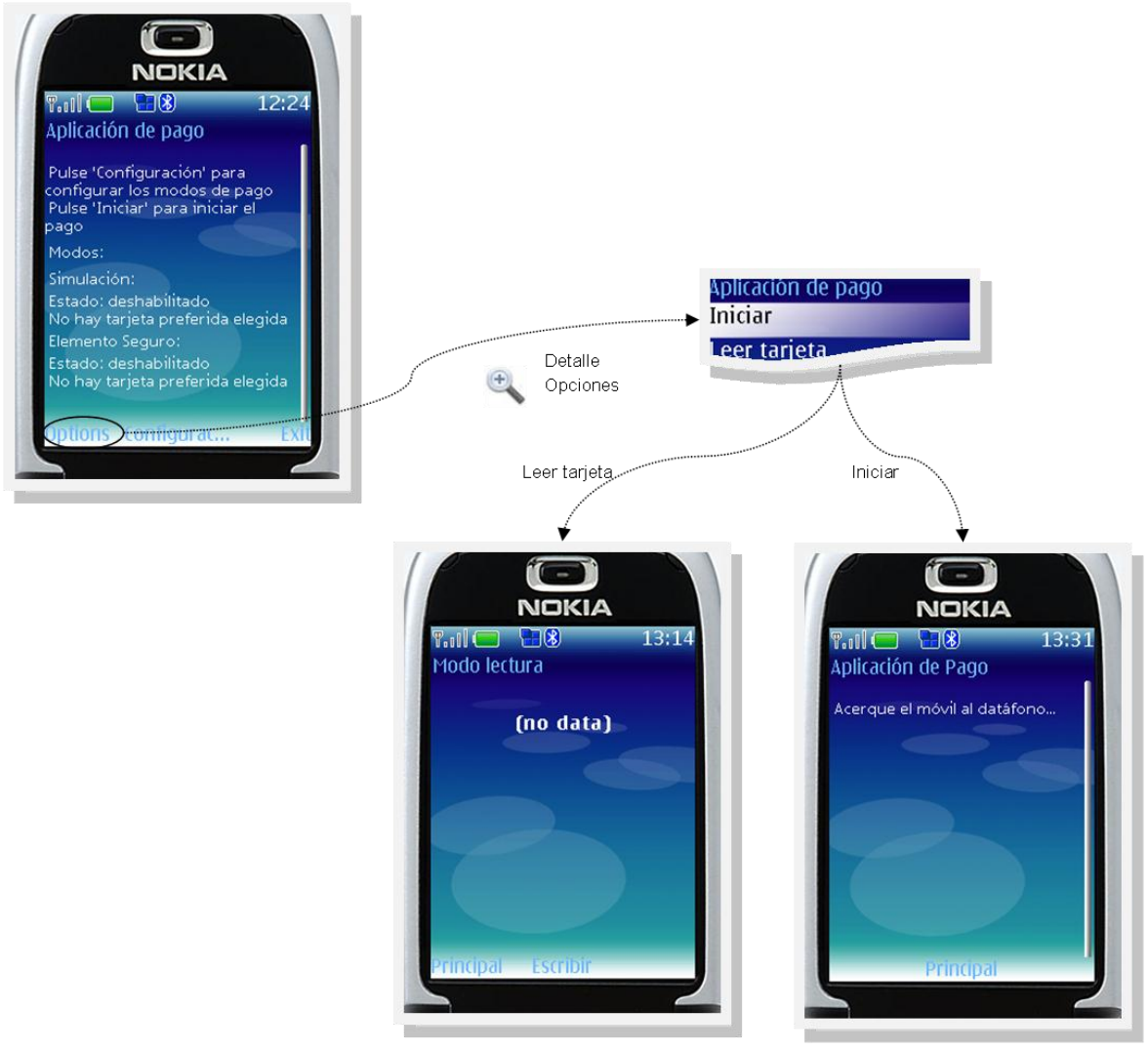

Figura 51: Uso de la aplicación móvil Cliente II

5.2 Aplicación móvil Reader

5.2.1 Diagrama de componentes

El diagrama de la Figura 52 muestra el paquete de la aplicación Reader, que se ejecuta en el teléfono móvil que funciona como Lector NFC. Esta aplicación se compone de tres clases Java:

- ReaderMIDlet: el *MIDlet* principal, donde se gestiona la comunicación y el funcionamiento principal, y la interfaz gráfica de la aplicación.
- RConexion: Clase que sirve para gestionar la comunicación NFCIP.
- Datafono: Clase que sirve para gestionar la comunicación Bluetooth.

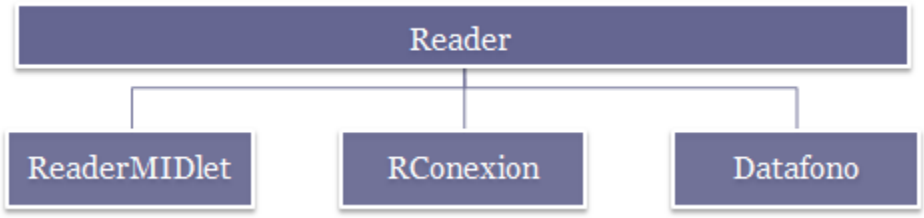

Figura 52: Diagrama de componentes de la aplicación Reader

5.2.2 Diagrama funcional de clases

El diagrama funcional de clases de la aplicación móvil Reader es el siguiente:

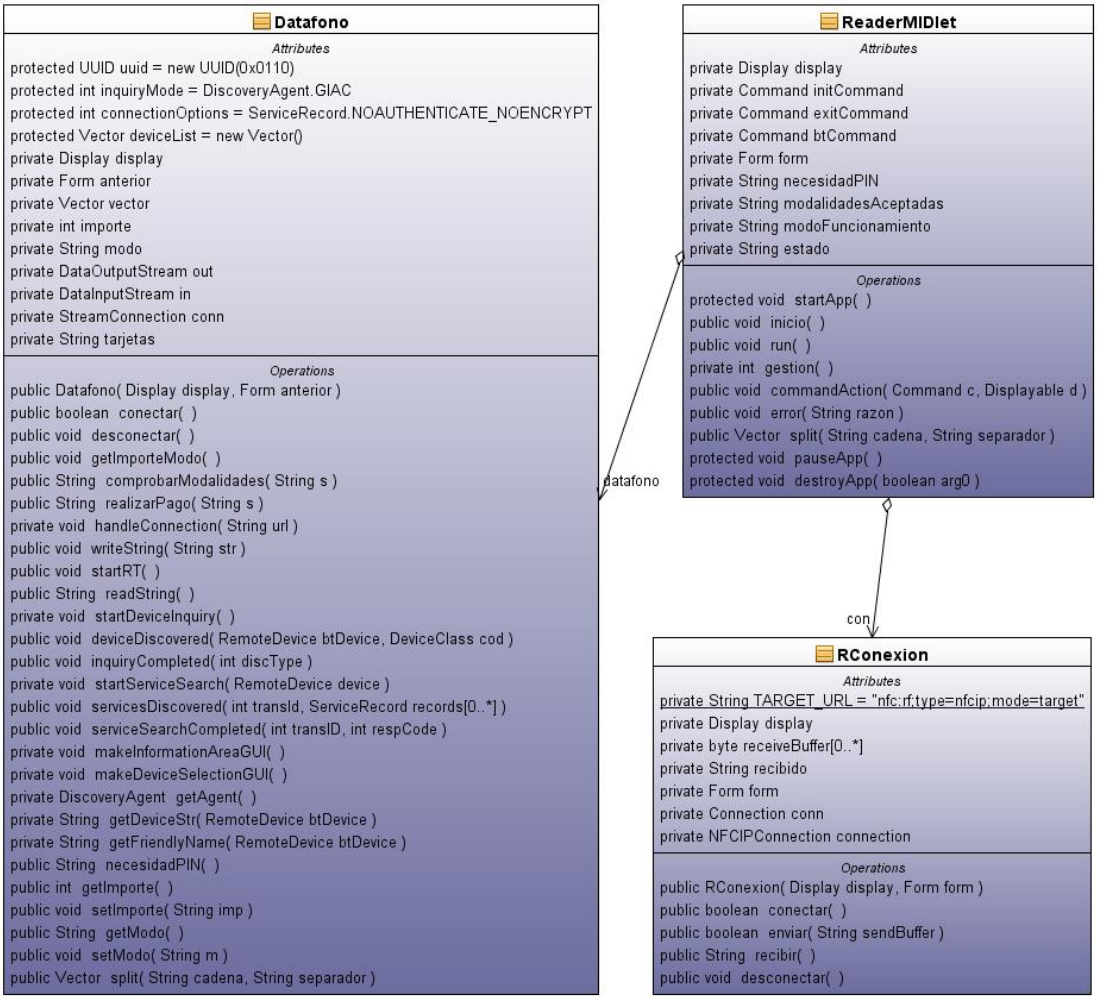

Figura 53: Diagrama funcional de la aplicación Reader

La clase principal ReaderMIDlet implementa un objeto de la clase RConexión, para poder realizar la comunicación NFCIP con el móvil cliente, y un objeto Datafono, para poder gestionar la comunicación Bluetooth con el PC.

5.2.3 Manual de usuario

A continuación se detalla cómo ha de manejarse la aplicación móvil Reader, por parte del empleado de la tienda.

La aplicación móvil Reader, ha de ser inicializada por el empleado de la tienda en la que se ofrece la posibilidad de realizar un pago a través de NFC. Además, se debe de tener a la vez inicializada y configurada la aplicación “PCServer” desde el ordenador de la misma tienda.

En el menú principal de la aplicación, como se observa en la siguiente Figura 54, se muestra la siguiente información:

- Pulse “Conexión Admin” para recibir los datos de pago desde el PC
- Pulse “Iniciar” para realizar el pago con el Cliente”.

Se debe, por tanto, acceder primero al menú “Conexión Admin” para poder realizar la conexión Bluetooth con el ordenador de la tienda, y así establecer conexión con el mismo y, por tanto, con la red bancaria.

Figura 54: Uso de la aplicación móvil Reader

En la figura, se puede observar que si se elige este menú, se informa en la pantalla de que se está realizando la conexión Bluetooth. El móvil, al comportarse como un cliente Bluetooth, muestra una lista de dispositivos con la que poder realizar conexión. Se deberá de elegir el dispositivo PC de la tienda. Al finalizar esta gestión, estará abierta una conexión Bluetooth entre el dispositivo PC de la tienda y el móvil Lector NFC de la tienda.

Seguidamente, se deberá de pulsar “Iniciar” para que el móvil lector se comporte como lector, es decir, esté a la escucha, esperando a que el móvil cliente se acerque para establecer una conexión NFCIP con él.

5.3 Aplicación PCServer

5.3.1 Diagrama de componentes

El diagrama de la Figura 55 muestra el paquete de la aplicación PCServer, que se ejecuta en el PC principal, funcionando como aplicación gráfica principal y como pasarela a la base de datos. Esta aplicación se compone de tres clases Java:

- PCServer: Clase principal, donde se gestiona la comunicación Bluetooth.
- VentanaCompra: Clase que sirve para modelar la interfaz gráfica de la aplicación a través de *Swing.*
- ConexionBancaria: Clase que sirve para gestionar la base de datos de la RedBancaria.

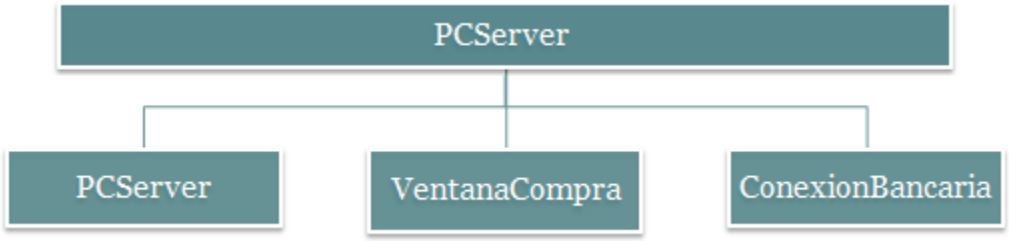

Figura 55: Diagrama de componentes de la aplicación PCServer

5.3.2Diagrama funcional de clases

El diagrama funcional de clases de la aplicación PCServer es el siguiente:

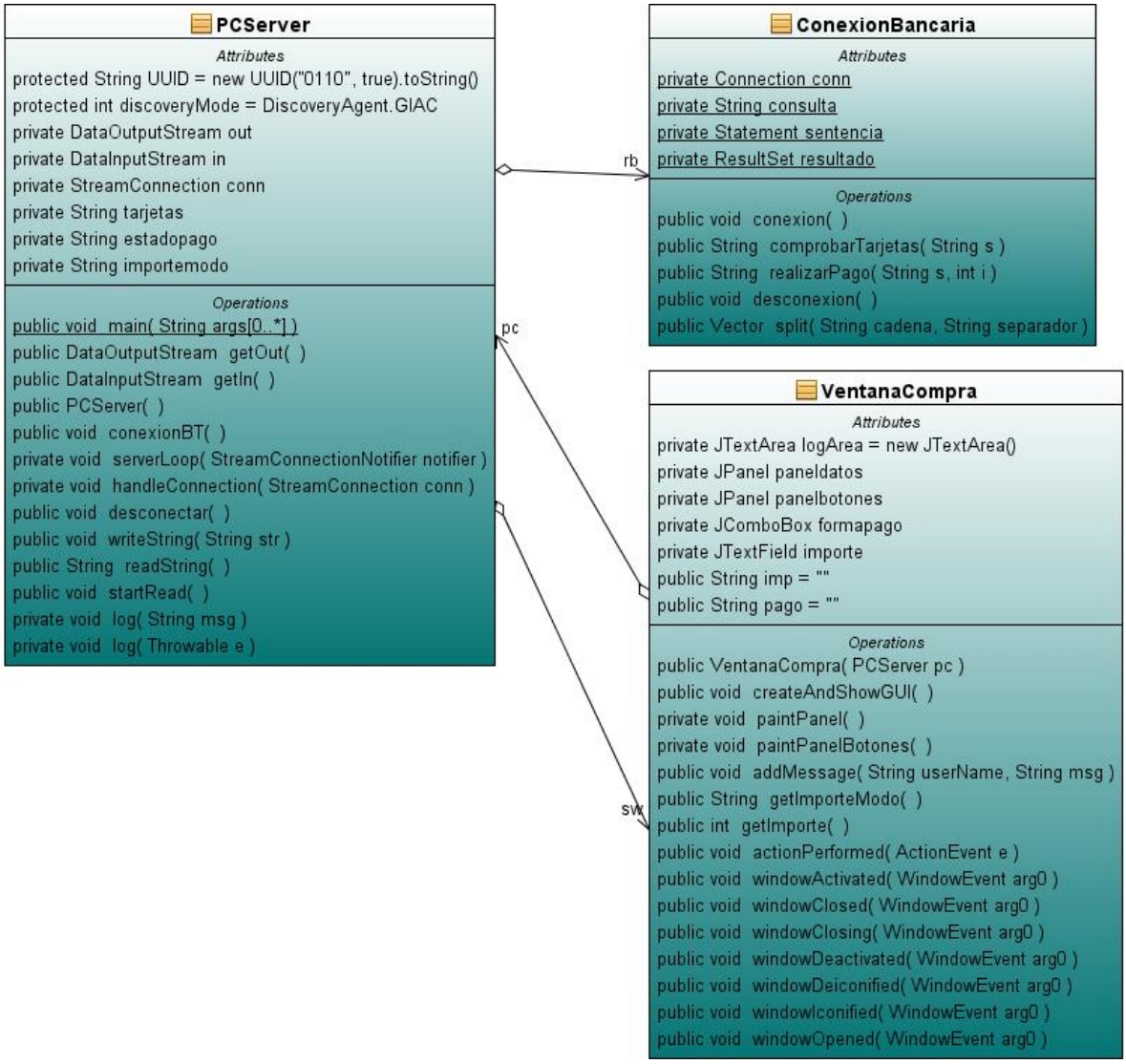

Figura 56: Diagrama funcional de la aplicación PCServer

En esta aplicación tenemos dos conexiones: la conexión Bluetooth, que es gestionada desde la clase principal PCServer, y la conexión a la base de datos, que es gestionada desde la clase ConexionBancaria. A través de la clase VentanaCompra se crea la interfaz de usuario gráfica.

5.3.3 Manual de usuario

A continuación se detalla cómo ha de utilizarse la aplicación PCServer de la tienda, manejada por el empleado de la tienda.

El empleado de la tienda ha de inicializar la aplicación gráfica PCServer en el ordenador de la tienda.

Deberá introducir en los campos correspondientes los siguientes datos: El importe de la compra, y la forma de pago elegida (Simulación o Elemento Seguro). Después se pulsará el botón “Iniciar conexión” para establecer una conexión Bluetooth con el teléfono móvil Reader NFC.

En ese momento, el estado de la aplicación será el mostrado en la siguiente Figura 57:

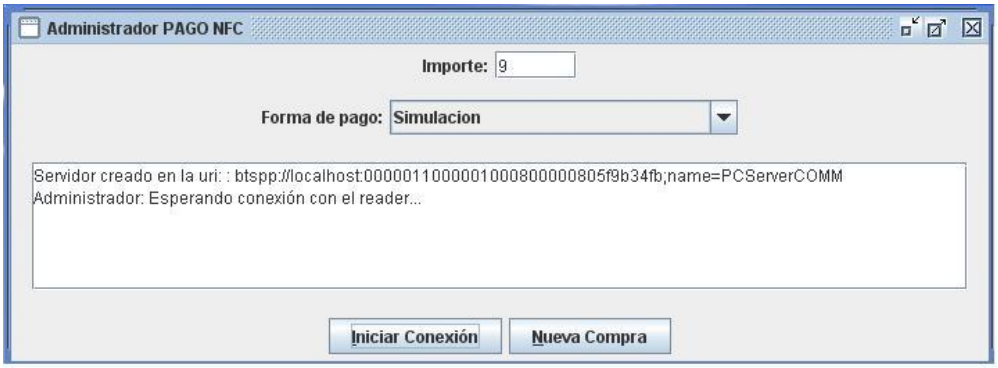

Figura 57: Uso de la aplicación PCServer I

Cuando se establezca la conexión Bluetooth con el lector, obtendremos lo siguiente:

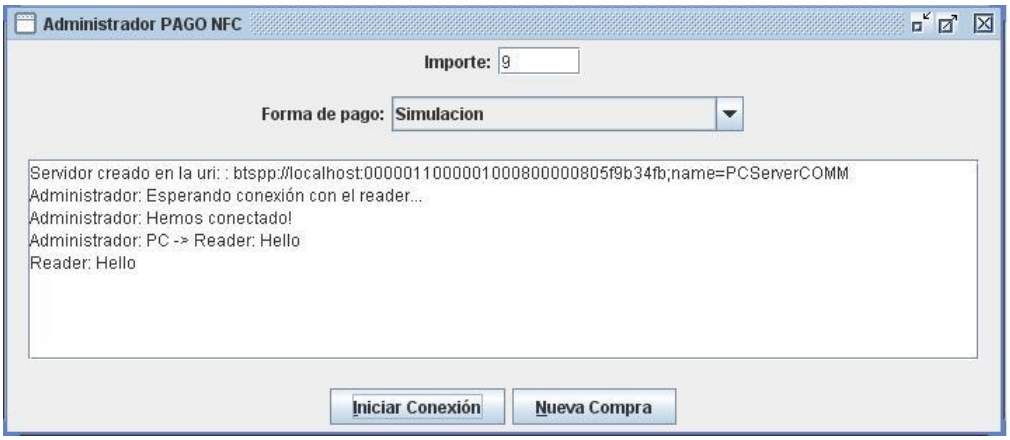

Figura 58: Uso de la aplicación PCServer II

Como se puede observar, en la línea de comandos de la aplicación, se pueden ver los mensajes intercambiados entre los distintos dispositivos del sistema.

Por ejemplo, en este punto, podemos observar que el Administrador, es decir, el PC Servidor, nos avisa de que hemos conectado. Seguidamente, el Administrador

muestra la trama que manda el PC al teléfono Reader, que en este caso es una trama de *handshake* "*Hello*". También se muestra la trama que nos llega desde el teléfono Reader, "*Hello*".

En este momento, el sistema de la tienda estará esperando a que el móvil cliente sea acercado al móvil lector para realizar el proceso de pago.

En la siguiente Figura 59, descrita posteriormente, se puede observar lo que sería mostrado por pantalla al realizar el proceso de pago:

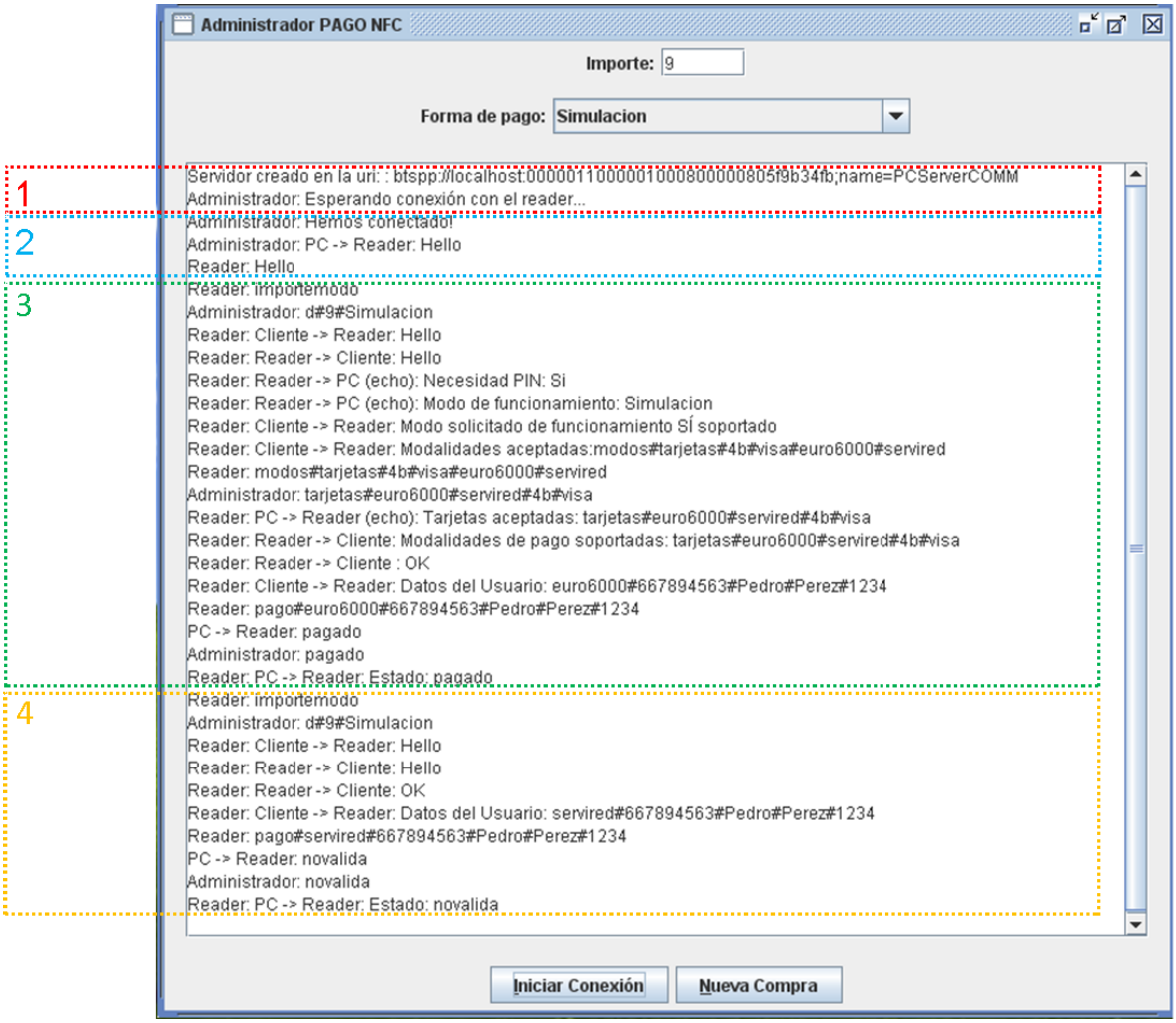

Figura 59: Uso de la aplicación PCServer III

El recuadro 1, muestra la situación inicial de espera.

El recuadro 2, muestra los mensajes iniciales correspondientes con la conexión Bluetooth.

El recuadro 3, engloba los mensajes correspondientes con la realización de un pago exitoso. En él se puede ver los mensajes intercambiados entre el Administrador, el móvil Reader y el móvil Cliente.

El recuadro 4, engloba los mensajes correspondientes con otro pago realizado por el mismo cliente pero con una tarjeta que está caducada.

El protocolo de comunicación seguido en este intercambio de tramas es descrito en el apartado 5.5.

El botón “Nueva compra” se debe de pulsar después de cambiar los parámetros de pago, para realizar una nueva transacción con datos distintos a los usados anteriormente.

5.4 Aplicación web RedBancaria

5.4.1 Diagrama de componentes

El diagrama de la Figura 60 muestra el paquete de la aplicación web RedBancaria.

Dentro del paquete Web, se encuentra:

- META-INF: Carpeta que incluye el archivo de configuración context.xml.
- WEB-INF: Carpeta que incluye el archivo de configuración web.xml, y la carpeta classes, que contiene los ficheros compilados (servlets, beans, etc.) de las clases utilizadas por la aplicación web contenidas en la carpeta “source”.
- img y style: Carpetas que guardan las imágenes y los archivos css, respectivamente.
- archivos JSPs.
-

Dentro del paquete Source, se encuentra:

- Beans: Carpeta que contiene las clases java beans.
- Database: Carpeta que contiene la clase java que permite las consultas a la base de datos.
- Servlets: Carpeta que contiene a los archivos Servlets.

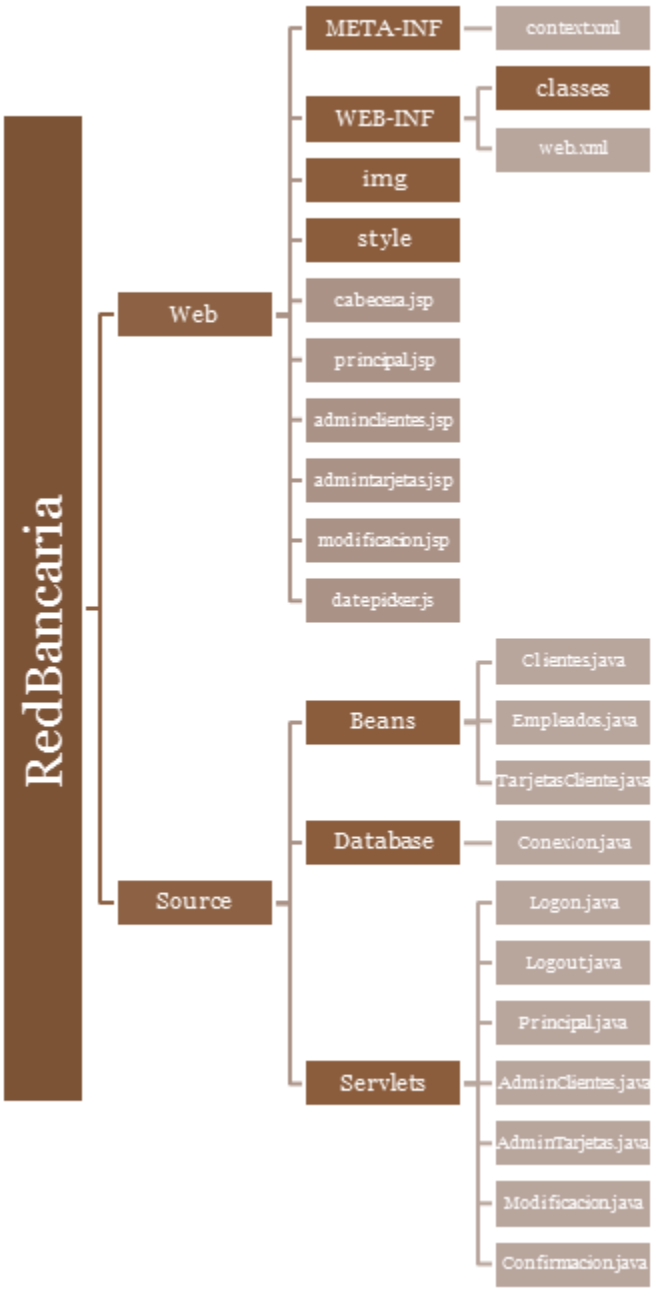

Figura 60: Diagrama de componentes de la aplicación web RedBancaria

5.4.2 Diagrama funcional de clases

El diagrama funcional de clases de la aplicación PCServer, se representa en dos partes.

En el primer diagrama, se observan las clases Java del proyecto: las clases Bean que representan a los clientes, tarjetas de los clientes y empleados, la clase Conexión, que gestiona la conexión con la base de datos, y las clases Servlets.

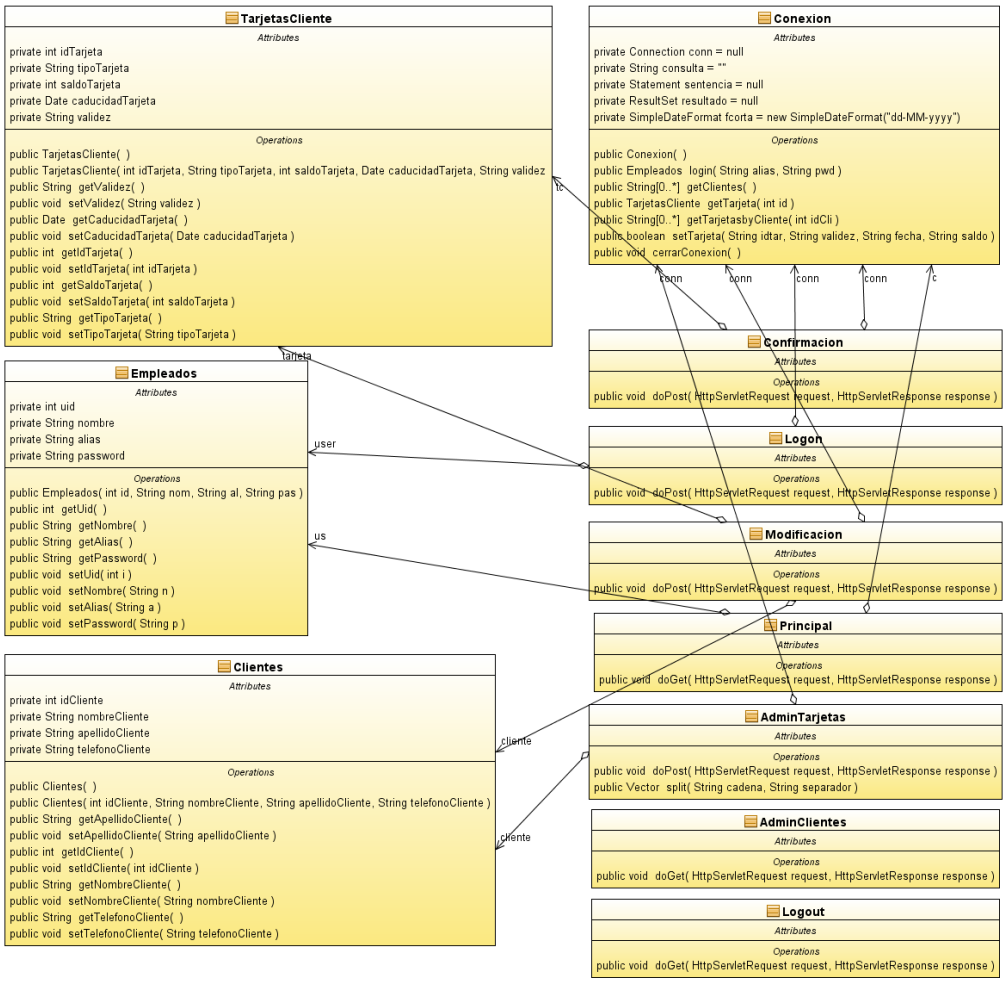

Figura 61: Diagrama de clases Java de la aplicación web Red Bancaria

En este segundo diagrama, se muestra la relación entre los servlets, que sirven para procesado de la información, y los JSPs, que se encargan de la presentación de dicha información. En este caso las fechas son de estado, no de dependencia.

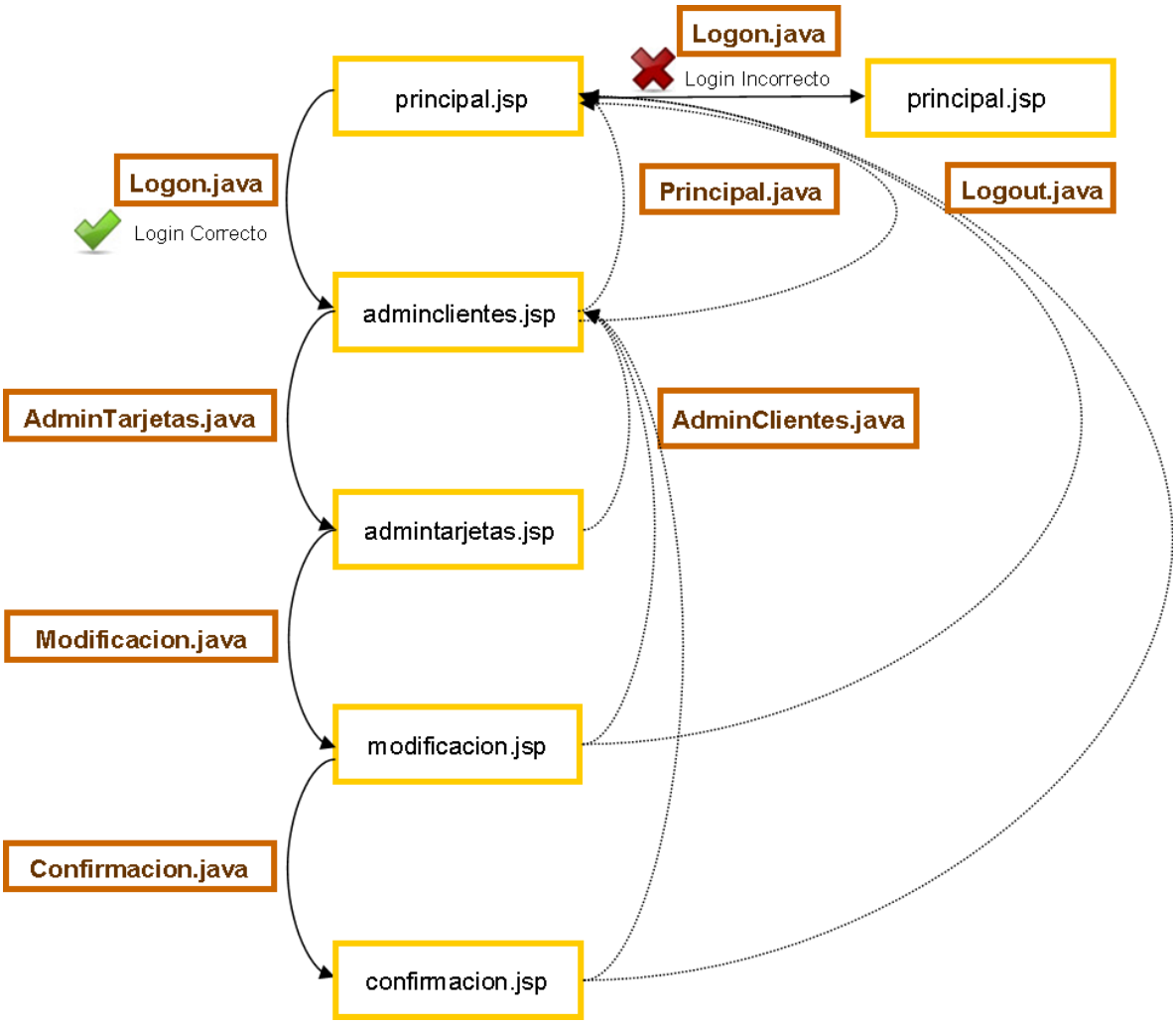

Figura 62: Diagrama funcional de la integración de los Servlets con los archivos JSPs de la aplicación web Red Bancaria

5.4.3 Manual de usuario

A continuación se describe cómo ha de utilizarse la aplicación web RedBancaria alojada en el servidor del banco y manejada por un empleado del banco.

El empleado del banco ha de utilizar esta aplicación web, cuando se requiera consultar o modificar los datos bancarios de los “clientes NFC” del banco.

La guía de la aplicación web se detalla en la siguiente Figura 63.

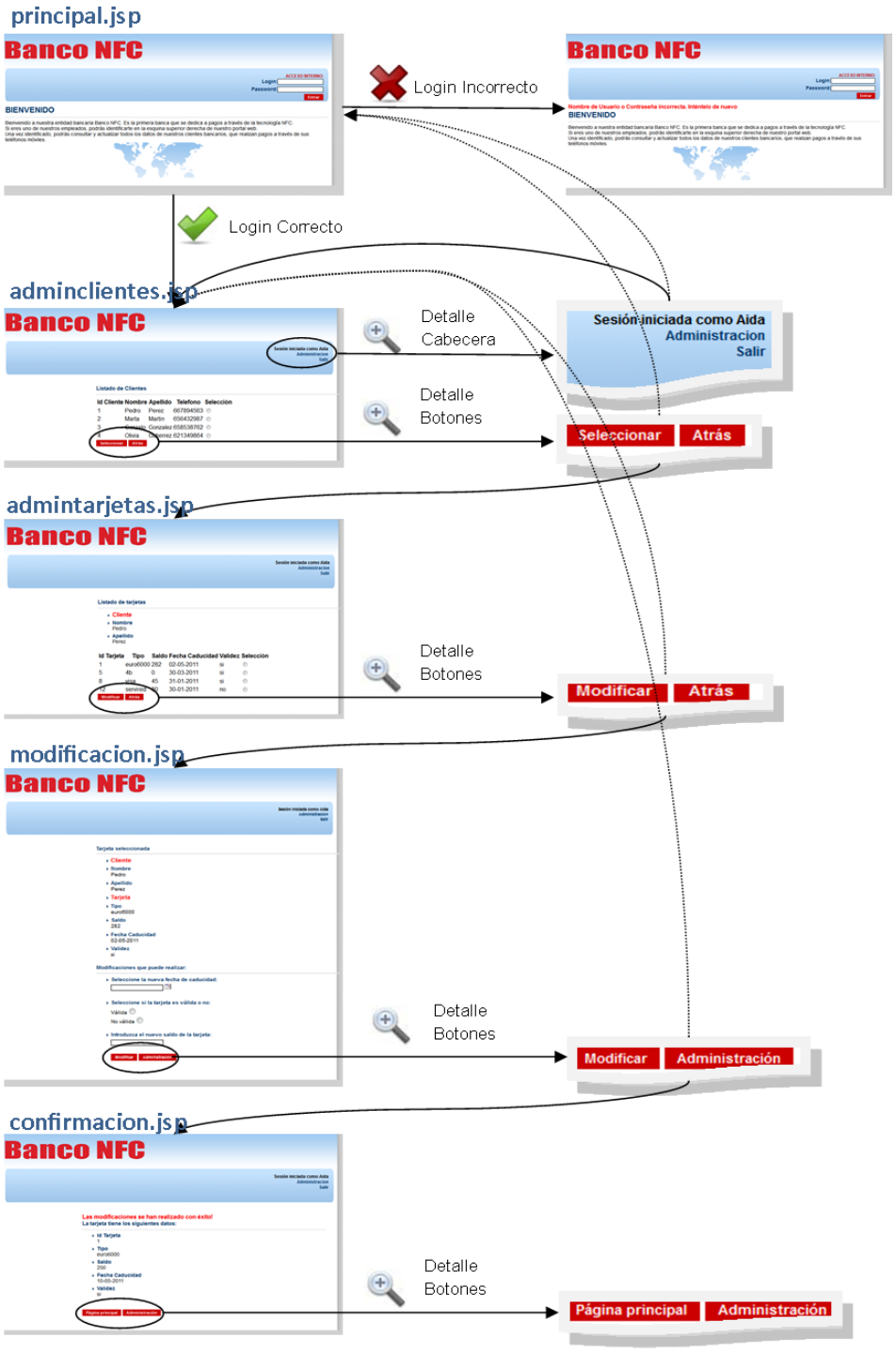

Figura 63: Uso de la aplicación web Red Bancaria

Desde la portada de la web, el empleado del banco se puede identificar rellenando los datos del formulario de la esquina superior derecha.

Al ser autenticado, el empleado podrá acceder al listado de Clientes en cuestión. Si selecciona uno, podrá acceder al listado de las tarjetas de ese cliente. Se mostrará en ese listado las características de cada tarjeta: su número de identificación, el tipo de tarjeta, su caducidad, su saldo y su validez.

Si se selecciona una, en el siguiente menú, se podrá modificar alguna o todas las características variables de la tarjeta: su saldo, su caducidad y su validez.

Si se pulsa al botón de confirmación “Modificar”, se mostrará los datos recién introducidos de la tarjeta modificada. Desde esa misma página y desde las anteriores, se puede retroceder al menú de Clientes.

5.5 Protocolo de comunicación diseñado

En este apartado se describe de forma detallada el protocolo de comunicación seguido entre los distintos elementos del sistema: el móvil cliente del usuario Cliente, el móvil Lector y el PC de la tienda, y la Red Bancaria.

Los elementos que forman parte de este protocolo se muestran en la siguiente figura:

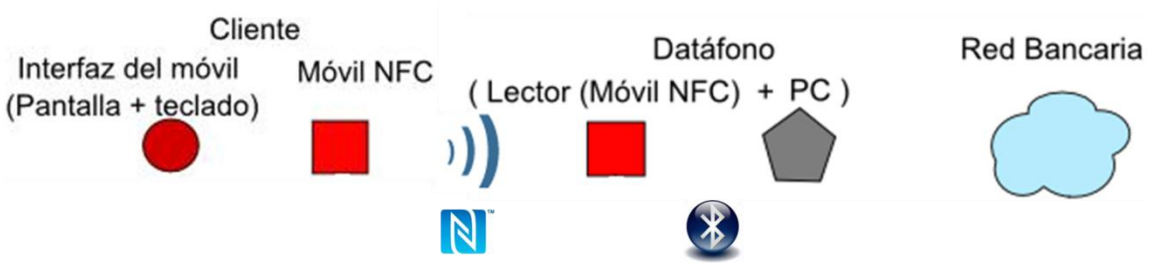

Figura 64: Elementos que componen el protocolo diseñado.

Este protocolo se podrá seguir de una forma visual más adelante. A continuación, se detalla la gestión que ha de realizar cada usuario cliente y empleado tienda. Después, se verá el intercambio producido en el sistema completo.

5.5.1 Lado Cliente

1. El cliente deberá habilitar un modo de funcionamiento.

2. Dentro de ese modo, puede elegir una tarjeta con la que desearía realizar el pago, a la que se llama tarjeta “preferida” o “predefinida”.

3. Deberá introducir un PIN de Tarjeta.

4. Deberá elegir la opción “Iniciar” para iniciar el pago, acercando su móvil al terminal Lector de la tienda.

5.5.2 Lado Tienda

1. El dependiente empleado de la tienda deberá también introducir el importe total del pago a realizar, así como el modo de funcionamiento en el que desea trabajar en la aplicación PCServer.

2. El dependiente deberá de establecer conexión Bluetooth entre el móvil Lector y el PC.

3. A continuación, el lector NFC se encontrará en estado de espera.

5.5.3 Sistema completo

1. Al lector se le acerca el móvil NFC del cliente. En este momento se produce una Conexión NFCIP y se inicia el Periodo de gestión.

2. *Handshake* (saludo inicial): Se manda “*Client_Hello*” por parte del móvil del usuario y “*Reader_Hello*” por parte del lector.

3. Si no fuera la primera conexión realizada por el cliente al datafono, debido a que ya ha hecho otro pago anteriormente, se comprueba si es necesario realizar alguna gestión preliminar antes de pasar directamente al Periodo de pago.

4. El datáfono dispone de un filtro según el precio del producto. Si éste supera una cierta cantidad determinada, es necesario el PIN del cliente. Para cantidades inferiores, no es necesario.

   Conociendo el precio del producto, el datáfono determinará si es necesario el PIN de la tarjeta que se usará a pagar, y le mandará esta información al móvil cliente, que la guardará.

   Si la tarjeta predefinida fuera aceptada, poseemos el PIN de la tarjeta desde el principio por lo que la información de la necesidad del PIN o no, no será imprescindible. Sí que la usaremos para que, si la tarjeta predefinida no es

aceptada, al elegir otra, saber si es necesario o no introducir el PIN de la nueva tarjeta.

También se mandará al móvil del cliente el modo de funcionamiento elegido al principio por el administrador del PC, para que sepa si tiene que trabajar en modo simulación o con el elemento seguro.

5. En el caso en el que el modo de funcionamiento requerido no esté soportado por el móvil del cliente, éste se lo comunicará al datáfono y se cerrará la conexión NFCIP. El lector quedará nuevamente a la espera. El cliente deberá modificar este dato y volver a establecer una conexión NFCIP, saltando al paso 1.

6. En el caso en el que modo de funcionamiento esté soportado, el cliente mandará un “OK” y todas las modalidades de pago aceptadas por él, al datáfono. Éste, a través de la red bancaria, comprobará cuales de ellas soporta, y se lo comunicará al móvil del cliente.

7. El móvil del cliente verá si la tarjeta predefinida se encuentra entre las modalidades de pago soportadas por el datáfono, o si, por el contrario, el cliente no le ha facilitado esta información Aquí, por tanto, habrá tres posibilidades:

    i. Que la tarjeta predefinida sea aceptada. (Se salta al paso 12)

    ii. Que la tarjeta predefinida no sea aceptada. (Se sigue en el paso 8)

    iii. Que la tarjeta predefinida no haya sido previamente seleccionada (Se sigue en el paso 8)

8. Si la tarjeta no es aceptada o no fue seleccionada anteriormente, el móvil cliente mostrará un aviso para que el cliente aparte su móvil del lector, por lo que se cerrará la conexión NFCIP. El lector quedará nuevamente a la espera.

9. El cliente deberá de elegir una tarjeta de las que le aparecen en el modo seleccionado.

10. En el caso en el que esta compra necesita del PIN de la tarjeta, el usuario deberá introducir el PIN de la tarjeta.

11. El cliente vuelve a acercar su móvil al datáfono para realizar otra conexión NFCIP.
12. Llegados este punto se inicia el Periodo de pago. A este periodo se entra:
    i. Si el usuario anterior tuvo que reelegir otra tarjeta si su predefinida no fue válida o si no la eligió al principio del proceso, y volvió a acercar el móvil al datáfono.
    ii. Si la tarjeta predefinida fue aceptada.
    iii. Si el mismo usuario anterior realiza otro pago.
13. Estamos en el Periodo de pago. Mandamos al datáfono nuestros datos de pago, incluyendo el PIN si fuera necesario.
14. El datáfono en este momento con toda la información necesaria, conecta con la red bancaria para realizar el pago, y ésta le devuelve un resultado.

Con la información obtenida de la red bancaria sobre el estado de pago, se le muestra al cliente a través de la pantalla de su móvil la acción realizada con el banco.

La red bancaria puede contestarle con la siguiente información:

1. Tarjeta no válida: Por algún motivo bancario (por ejemplo, por seguridad), la tarjeta no es válida, por lo que el pago no puede ser realizado.
2. Tarjeta caducada: La red bancaria ha comprobado que la tarjeta es válida pero está caducada, por lo que no se puede realizar el pago.
3. Tarjeta sin saldo: La red bancaria ha comprobado que la tarjeta que ha decidido utilizar el cliente, a pesar de ser válida, no tiene saldo suficiente para realizar la compra. Al usuario se le informa de que su tarjeta no tiene saldo.
4. Tarjeta válida, pago realizado: La red bancaria ha podido realizar satisfactoriamente el pago con el emisor de la tarjeta y le informa al usuario de que la transacción ha sido aceptada.

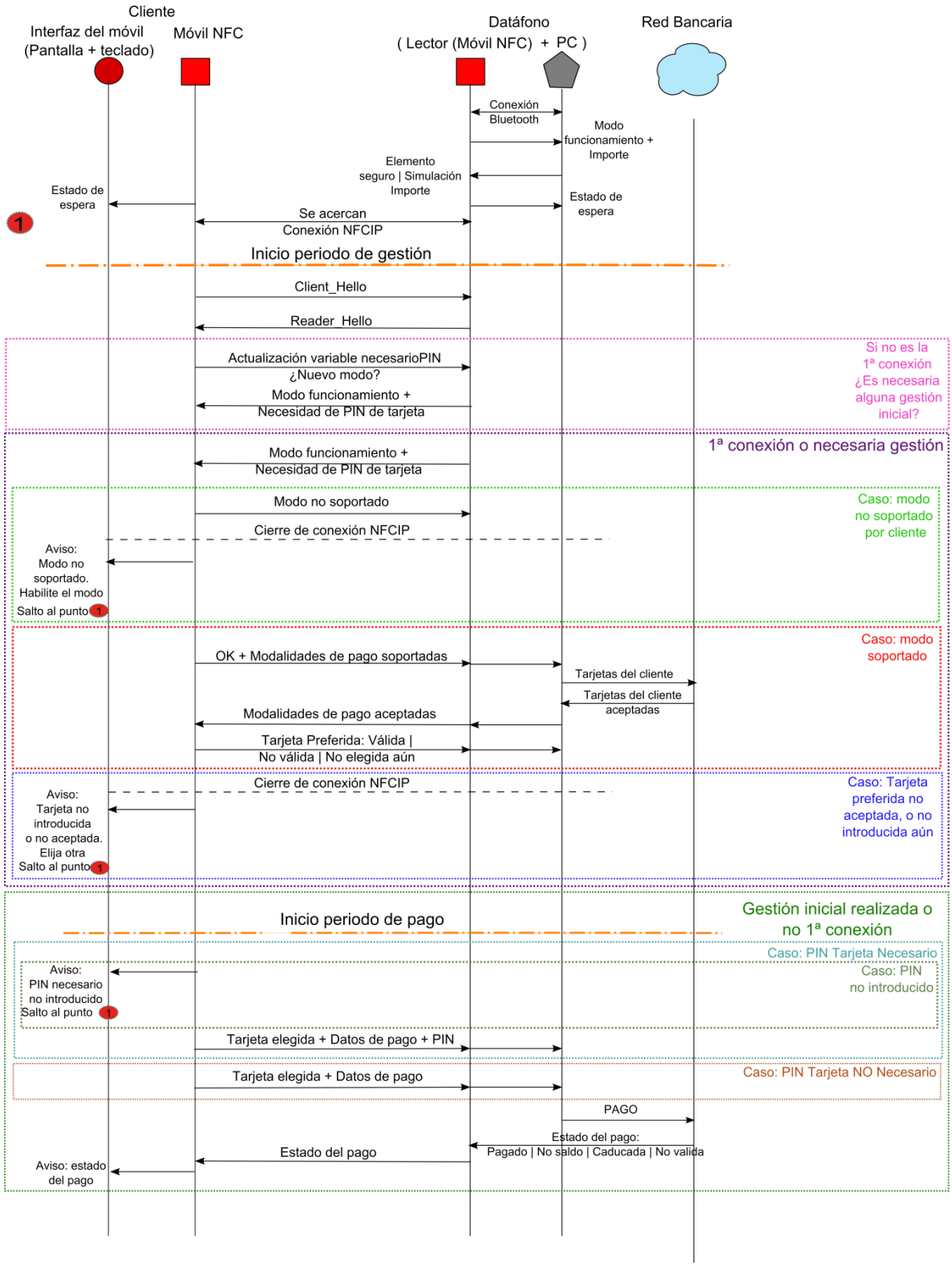

Figura 65: Protocolo de comunicación diseñado

Capítulo 6 : Pruebas realizadas

En el presente capítulo se recogen distintas pruebas que se han realizado a lo largo del desarrollo de este proyecto para poder validar las funcionalidades y objetivos que se han marcado tanto en el planteamiento, como en el propio desarrollo.

6.1 Aplicaciones

A continuación, se recogen las pruebas realizadas a la funcionalidades de cada aplicación.

6.1.1 Aplicación móvil Cliente

- Elección de una tarjeta cuando es conocido que es requerido el PIN: Si el PIN no es introducido o es incorrecto, no se puede elegir esa tarjeta. En ese caso se mostrará un mensaje de alerta “Pin Incorrecto”. Si no es necesario el PIN, acepta elección de una tarjeta sin haberlo introducido.

Figura 66: Ejemplo gráfico de mensaje de alerta en el dispositivo móvil informando de un error

- Correcto funcionamiento en la configuración y gestión de modos de funcionamiento y tarjetas: Se almacenan correctamente los datos seleccionados

- Actualización en el modo de Simulación de los datos bancarios del cliente al acercar una tarjeta externa. Si no se ha acercado la tarjeta externa, el modo Simulación aparece sin tarjetas bancarias posibles.

Figura 67: Modo Simulación sin estar configurado a través de la tarjeta NFC externa

- Si se pulsa “Iniciar el pago”, se queda en modo espera y sugiere al cliente que acerque el móvil al lector, iniciándose el protocolo de comunicación una vez conectado con el móvil lector.

6.1.2 Aplicación móvil Reader

- Correcto funcionamiento y gestión de ambas comunicaciones: NFCIP y Bluetooth. En caso de error, se sugiere el reinicio de la aplicación por pantalla.

6.1.3 Aplicación PCServer

- Control de los parámetros introducidos por el empleado de la tienda por pantalla. Si no se introduce alguno, se muestra error por pantalla al intentar iniciar la conexión Bluetooth con el Lector.

- Aparición de todas las tramas que conlleva el proceso de comunicación sobre el protocolo definido, apareciendo si hay error en la comunicación, su motivo.

- Botón nueva compra: utilizado para el almacenamiento de los nuevos parámetros en caso de cambiar los mismos para un mismo cliente.

6.1.4 Aplicación web Red Bancaria

- Correcto funcionamiento y conexión con la base de datos.

- Control de errores en el proceso de autenticación por parte del empleado bancario.

- Informe de error al realizar modificaciones en alguno de los datos bancarios, si el proceso ha sido fallido o no ha podido completarse.

6.2 Comunicación

A continuación, se recogen las pruebas realizadas al proceso de comunicación.

- Dispositivo Reader o dispositivo móvil son separados durante su comunicación: Se obtiene un mensaje de error en los móviles indicando que ha habido un error en el protocolo.

- Error en dispositivo Cliente “Modo no soportado”: Este caso se da cuando el empleado de la tienda de nuestro sistema simulado exige para la operación

un modo de funcionamiento no habilitado por el dispositivo móvil cliente. Este error es avisado al cliente a través de un aviso de alerta: “Modo solicitado por el Reader no soportado”. En la pantalla de la aplicación gráfica de la tienda aparecerá la siguiente trama:

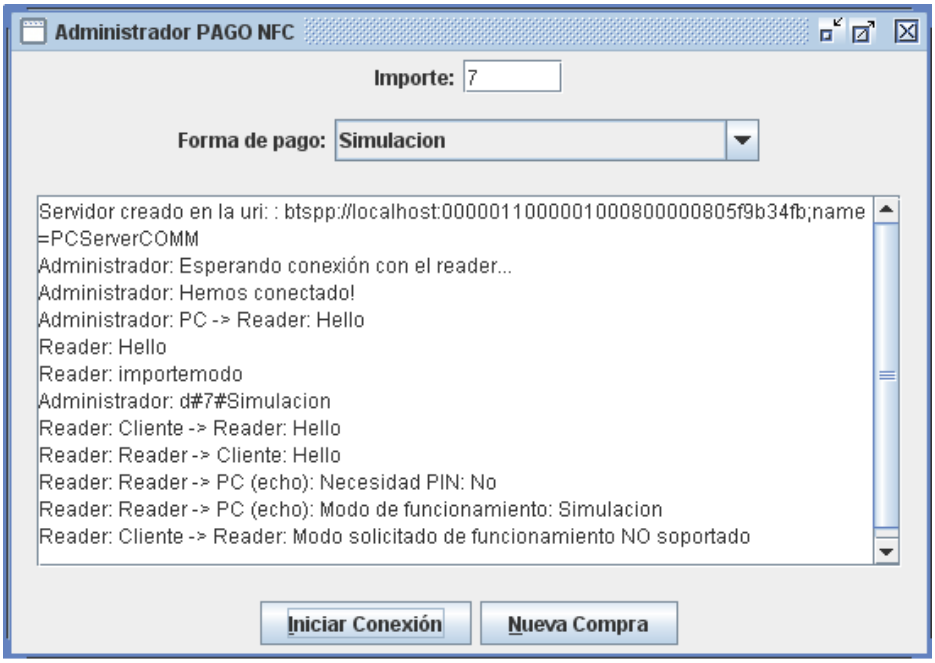

Figura 68: Error “Modo no soportado” mostrado en la aplicación gráfica de la tienda.

- Error en el dispositivo Cliente “Tarjeta no válida, no seleccionada”: Si la tarjeta no es aceptada o no fue seleccionada anteriormente, el móvil cliente mostrará un aviso para que el cliente aparte su móvil del lector.

- Actualización correcta de las tarjetas soportadas por la red Bancaria en el listado de tarjetas en el dispositivo móvil cliente. Las tarjetas que tiene en cliente configuradas en su móvil se comparan con las soportadas por la red Bancaria, de tal forma que se filtran sólo las que soporta, dando posibilidad únicamente de elección de las soportadas.

- Se ha realizado una batería de pruebas realizando varias conexiones con el mismo cliente con distintas configuraciones. Algunas de ellas, como ejemplo, son:

    - Primera conexión: modo no soportado.
      Segunda conexión: tarjeta no válida.
      Tercera conexión: tarjeta válida.

  - Primera conexión modo no soportado
    Segunda conexión: tarjeta válida

  - Primera, segunda y tercera conexión: tarjeta válida

  - Selección de tarjeta válida en modo no soportado, e intentando realizar pago con modo sin configurar.

  - Conexiones seguidas para distintos estados de pago bancarios: tarjeta caducada, no válida, sin saldo.

- Error en el dispositivo cliente si se realiza pago con tarjeta sin PIN, para un importe que requiera pago online. Se muestra el aviso en el móvil cliente: “Es necesario PIN”

Capítulo 7 : Conclusiones y trabajos futuros

En el presente capítulo se desarrollan, en un primer apartado, las conclusiones derivadas al desarrollo de este proyecto. En el siguiente apartado, se realizará una visión de futuro respecto a las posibles líneas en las que poder seguir trabajando.

7.1 Conclusiones

En este proyecto se ha desarrollado un sistema simulado completo de un escenario de pago bancario, a través del uso de distintas tecnologías de conexión, centrándose en la tecnología NFC.

- El desarrollo de este proyecto se ha realizado parcialmente bajo las direcciones de la empresa “Gamma Solutions”, tomando parte de las decisiones iniciales del diseño.

- Se ha podido constatar que la tecnología NFC está en pleno auge, encontrándose indicios que apuntan a que esta tecnología puede llegar a triunfar, extenderse y consolidarse con el paso de los años. La gran cantidad de información sobre NFC que se puede encontrar en la web

actualmente, dista mucho de la que se podía encontrar en el comienzo del desarrollo de este proyecto.

- El caso de uso de la tecnología NFC estudiado en este proyecto, un sistema de pago móvil, es al que más importancia se le está dando en el mercado, realizándose pruebas piloto de aplicaciones semejantes a la desarrollada en este proyecto por todo el mundo.

- La programación usando la tecnología Bluetooth ha sido bastante costosa debido a problemas de software y hardware, teniendo que cambiar de dispositivo USB Bluetooth, e incluso de Sistema Operativo varias veces.

- Se ha conseguido realizar un sistema simulado completo de un escenario de compra habitual, que puede ser realizado en cualquier tienda, teniendo que tomar muchas decisiones de diseño y de simulación. Se ha intentado establecer un escenario realista, añadiéndole detalles técnicos para facilitar su entendimiento.

- Ha resultado ser muy interesante el desarrollo de este proyecto, debido a que ha posibilitado trabajar con la tecnología NFC, que está en pleno crecimiento, profundizar en el lenguaje de programación Java e integrar distintas tecnologías en un mismo entorno.

7.2 Trabajos futuros

En este apartado se analizan las posibles líneas de trabajo futuras. Estas pueden ser:

- Despliegue de seguridad en el sistema: La seguridad en un sistema de comunicación inalámbrico es un elemento clave en su utilización. Además, si las operaciones que vamos a realizar a través del canal son bancarias, para llevar a cabo un pago, la seguridad se convierte en un punto aún más importante. Esta sería la primera línea de trabajo futuro necesaria, el cifrado de los canales de comunicación y encriptación de los datos más críticos.

- Desarrollo del escenario con un Lector NFC real: En este proyecto se ha utilizado como lector NFC un dispositivo móvil Nokia. Una línea de futuro propuesta sería incorporar en el escenario el elemento del lector NFC, como dispositivo real.

- Mejora de la interfaz de la aplicación de la tienda. La aplicación de la tienda a la que accede el empleado, podría ser mejorada, tanto en su aspecto o diseño, como en su funcionalidad.

- Desarrollo de las aplicaciones para otros sistemas operativos. Actualmente en el mercado, están apareciendo dispositivos móviles *smartphones* con capacidades NFC, que son los dispositivos que realmente van a ser utilizados para los pagos en escenarios reales. Algunos de estos dispositivos tienen los sistemas operativos Blackberry O.S. o Android. Una línea de trabajo futuro muy interesante sería el desarrollo de las aplicaciones móviles para estos sistemas operativos.

- Mejora del sistema simulado de la Red Bancaria. O inclusive, adaptarlo para un entorno real empresarial cerrado.

Capítulo 8 : Presupuesto

En este capítulo se realizará un análisis del desarrollo de este proyecto en términos económicos.

8.1 Planificación de tareas

Para poder realizar un presupuesto del sistema expuesto, se tienen en cuenta las fases no cronológicas mostradas en la introducción del presente documento, en el apartado 1.3. En la siguiente tabla se muestran estas fases junto con el número de horas estimadas que se han dedicado a cada una de ellas.

Fase	Descripción	Número de horas
0	Investigación y documentación: NFC y Bluetooth	90 horas
1	Despliegue y configuración entorno de trabajo	10 horas
2	Estudio y diseño funcionalidad del sistema	100 horas
3	Desarrollo funcionalidades NFC	100horas
4	Desarrollo funcionalidades Bluetooth	80 horas
5	Integración funcionalidades NFC y Bluetooth	100 horas
6	Desarrollo y diseño Base de datos	20 horas
7	Desarrollo aplicación web	40 horas
8	Integración del sistema completo	100 horas
9	Realización de pruebas	60 horas
10	Redacción de la memoria	120 horas
11	Corrección de la memoria	10 horas

Tabla 9: Relación de horas dedicadas a las distintas fases que comprenden el desarrollo del proyecto

8.2 Costes

En este apartado se determinan los costes debido a los siguientes factores: el personal, el material (dividiéndolo en equipo y licencias) y los costes indirectos.

8.2.1 Personal

Para el desarrollo de este proyecto, se ha de tener en cuenta los distintos especialistas que realizarían las diferentes tareas presentadas anteriormente como fases, y el precio por hora trabajada. En la siguiente tabla se establecen los costes por hora de los especialistas que se consideran en la realización del proyecto.

Cargo	Coste por hora
Analista	35 €
Diseñador	40 €
Gestión calidad y pruebas	35 €
Programador	25 €

Tabla 10: Relación de los costes por hora de los distintos especialistas que intervienen en la realización de este proyecto

Por tanto, aplicando estos costes e identificando el especialista por tarea, se calcula el coste total de personal requerido para este proyecto, a través de la siguiente tabla:

	Fase	Cargo	Número de horas	Coste
0	Investigación	Analista	90 horas	3190 €
1	Entorno trabajo	Programador	10 horas	250 €
2	Diseño sistema	Diseñador	100 horas	4000 €
3	Desarrollo NFC	Programador	100 horas	2500 €
4	Desarrollo Bluetooth	Programador	80 horas	2000 €
5	Integración NFC y Bluetooth	Programador	100 horas	2500 €
6	Desarrollo Base de datos	Programador	20 horas	500 €
7	Desarrollo web	Programador	40 horas	1000 €
8	Integración sistema	Programador/Diseñador	100 horas	3500 €

9	Pruebas	Gestión calidad y pruebas	60 horas	2100 €
10	Redacción memoria	Analista/ Diseñador	120 horas	4500 €
11	Corrección memoria	Analista/Diseñador	20 horas	750 €
			Total	**26790 €**

Tabla 11: Costes del personal

8.2.2 Material

En este apartado se recogen los costes derivados por el material utilizado en el desarrollo de este proyecto, que podemos dividir en equipo y licencias.

8.2.2.1 Equipo

En la siguiente tabla se muestra una relación del equipo utilizado para el desarrollo de este proyecto. Se toma como coeficiente de amortización un tercio de su precio para cada dispositivo.

Dispositivo	Precio/unidad	Coeficiente amortización	Unidades	Precio total
Móvil Nokia 6131 NFC	190 €	1/3	2	127 €
Ordenador portátil	690 €	1/3	1	230 €
Tarjeta MIFARE 1k	0.90€	1/3	1	0.30 €
Dispositivo USB Bluetooth	12 €	1/3	1	4 €
			Total	**361.30 €**

Tabla 12: Costes de equipo

8.2.2.2 Licencias

En la siguiente tabla se recogen las licencias utilizadas, que han resultado ser gratuitas.

Concepto	Licencia	Coste/Licencia	Coste
Java SE Development Kit 6u16	1	0 €	0 €
NetBeans IDE 6.9	1	0 €	0 €
Nokia 6131 NFC SDK 1.1	1	0 €	0 €
Librería Bluecove 2.1.0	1	0 €	0 €
Inkscape 0.48	1	0 €	0 €
		Total	**0 €**

Tabla 13: Costes de licencias

8.2.3 Indirectos

En este apartado se presentan los costes indirectos, es decir, aquellos costes que se derivan del uso de instalaciones para el desarrollo del proyecto.
Se toma como tiempo estimado de realización del proyecto un año, y como coste indirecto el 20% de cada coste total por concepto al mes.

Concepto	Cantidad
Alquiler del local	1440 €
Luz y agua	960 €
Teléfono y conexión a internet	600 €
Total	**3000 €**

Tabla 14: Costes indirectos

8.3 Total

En la siguiente tabla establecemos el coste total a partir de los resultados obtenidos en el anterior apartado.

Concepto	Cantidad
Personal	26790.00 €
Equipo	361.30 €
Licencias	0 €
Costes indirectos	3000.00 €
Coste Total	**30151.30 €**

Tabla 15: Coste total

Por tanto, sumándole a esa cantidad: 10% que estimamos como Riesgo, 20% para Beneficio, y el 16% de I.V.A:

Concepto	Cantidad
Coste Total	**30151.30 €**
Riesgo (10 %)	**3015.13 €**
Beneficio (20 %)	**6030.26 €**
Total sin I.V.A	**39196.69 €**
I.V.A (16%)	**6271.47 €**
Total	**45468.16 €**

Tabla 16: Presupuesto total

El presupuesto total de ese proyecto, teniendo en cuenta los costes desglosados en los apartados anteriores, asciende a la cantidad de **CUARENTA Y CINCO MIL CUATROCIENTOS SESENTA Y OCHO CON DIECISÉIS** euros

Leganés, a 3 de Junio de 2011

El ingeniero proyectista

Fdo: Aida Campa Ruíz.

Glosario

APDU - Application Protocol Data Unit
API - Application Programming Interface
AWT - Abstract Window Toolkit
CDC - Connected Device Configuration
CLDC - Connected Limited Device Configuration
CMOS - Complementary metal-oxide-semiconductor
CORBA - Common Object Request Broker Architecture
CVM - Compact Virtual Machine
EDGE - Enhanced Data rates for GSM of Evolution
EJB - Enterprise Java Beans
ETSI - European Telecommunications Standards Institute
FTP - File Transfer Protocol
GNU GPL - GNU General Public License
GPRS - General Packet Radio Service
GUI - Graphical User Interface
HTML - HyperText Markup Language
HTTP - Hypertext Transfer Protocol
IDE - Integrated Development Environment
IEC - International Electrotechnical Commission
IP - Internet Protocol
ISM - Industrial, Scientific and Medical (band)
ISO - International Standarization Organization
J2EE - Java 2 Platform Enterprise Edition
J2ME - Java 2 Platform Micro Edition
J2SE - Java 2 Platform, Standard Edition
JAD - Java Application Descriptor
JAR - Java Archive
JDBC - Java Database Connectivity
JDK - Java Development Kit

JFC - Java Foundation Classes
JRE - Java Runtime Environment
JSP - Java Server Pages
JSR - Java Specification Requests
JVM - Java Virtual Machine
KVM - Kilo Virtual Machine
L2CAP - Logical Link Control and Adaptation Protocol
LAN - Local Area Network
LLCP - Logical Link Control Protocol
MIDI - Musical Instrument Digital Interface
MIDP - Mobile Information Device Profile
MIME - Multipurpose Internet Mail Extension
MMC - Multi Media Card,
MMS - Multimedia Messaging System
MS - Memory Stick
NDEF - NFC Data Exchange Format
NFC - Near Field Communication
NFCIP - Near-Field Communication Interface and Protocol
OBEX - Object Exchange
OSI - Open System Interconnection
OTA - Over-The-Air
P2P - Peer to Peer
PDA - Personal Digital Assistant
PIM - Personal Information Manager
QVGA - Quarter Video Graphics Array
RF - Radio Frequency
RFID - Radio Frequency Identification
RMI - Remote Method Invocation
RTD - Record Type Definition
SD - Secure Digital
SDK - Software Development Kit
SIG - Special Interest Group
SIM - Subscriber Identification Module
SIND - Speaker Independent Name Dialing
SMS - Short Message Service
SPP - Serial Port Profile
SQL - Structured Query Language
TFT - Thin Film Transistor
URI - Uniform Resource Identifier
WAR - Web Application Archive
WIFI - Wireless Fidelity
XHTML - Extensible Hypertext Markup Language
XML - Extensible Markup Language

Bibliografía

[1] NFC-Forum, “NFC-Forum whitepapers”, <*http://www.nfc-forum.org/resources /white_papers/>,* 2001 [Disponible en Internet, 27 de Abril 2011]

[2] Nokia, “Nokia NFC Discussion Forum”, <*http://discussion.forum.nokia.com /forum/*>, 2011 [Disponible Internet, 27 de Abril 2011]

[3] NFC Forum, “The NFC Forum”, <*http:/www.nfc-forum.org/aboutus/*>, 2011 [Disponible Internet, 27 de Abril 2011]

[4] C. Enrique Ortiz, Oracle, “An Introduction to Near-Field Communication and the Contactless Communication API”, <http://java.sun.com/developer/ technicalArticles/javame/nfc> ,Junio 2008 [Disponible Internet, 27 de Abril 2011]

[5] Rubén Abuín, Raúl de Benito, Universidad de Deusto, “Near field Communication”, <*http://www.slidefinder.net/n/nfc/8057837*>, Junio 2008 [Disponible Internet, 27 de Abril 2011]

[6]Carlos Cossio, “Tecnología NFC: Arquitectura Dispositivo móvil NFC”, <http://www.terra.es/personal/ccossio/tecnologiaNFC_8.htm>, 2008 [Disponible Internet, 27 de Abril 2011]

[7] Qusay H. Mahmoud, “Part II: The Java APIs for Bluetooth Wireless Technology” <*http://developers.sun.com/mobility/midp/articles/bluetooth2*>, Abril 2003 [Disponible Internet, 27 de Abril 2011]

[8] Alberto Gimeno Brieba, “JSR-82: Bluetooth desde Java™” <*http://mami.uclm.es/j2me/J2ME/java-bluetooth.pdf*>, 2004. [Disponible Internet, 27 de Abril 2011]

[9] Wikipedia, “Java”, <*http://es.wikipedia.org/wiki/Java_%28lenguaje_de_programaci%C3%B3n%29>,* Abril 2011 [Disponible Internet, 27 de Abril 2011]

[10] ProgramacionEnJava, “Tutorial: Características de Java”, <*http://www.programacionenjava.com/blog/2008/06/10/tutorial/tutorial-caracteristicas-de-java>,* Junio 2008 [Disponible Internet, 27 de Abril 2011]

[11] OsmosisLatina, “Java” <*http://www.osmosislatina.com/java/componentes.htm*> Febrero 2005, [Disponible Internet, 27 de Abril 2011]

[12] Oracle, “Java ME Technology APIs & Docs”, <*http://www.oracle.com/technetwork/java/javame/documentation/apis-jsp-137855.html*> [Disponible Internet, 27 de Abril 2011]

[13] David Díaz Martín, Antonio Hernández Serrano, “Introducción a J2ME”, <*http://zarza.fis.usal.es/~fgarcia/docencia/poo/01-02/trabajos/S3T1.pdf*>, 2002 [Disponible Internet, 27 de Abril 2011]

[14] Grupo de Investigación en Agentes Software: Ingeniería y Aplicaciones, Universidad Complutense de Madrid, “J2ME”, <*http://grasia.fdi.ucm.es/j2me/_J2METech/index.html*>, 2004 [Disponible Internet, 27 de Abril 2011]

[15] Luis E. Aponte I., “Interfaces gráficas de Usuario”, <*http://programandoenjava. over-blog.es/article-i-51475428.html>,* Junio 2010 [Disponible Internet, 27 de Abril 2011]

[16] Oracle, “The Swing Tutorial”, <*http://download.oracle.com/javase/tutorial/uiswing/index.html>,* 2010 [Disponible Internet, 27 de Abril 2011]

[17] Wikipedia, “Java Servlet” <*http://es.wikipedia.org/wiki/Java_Servlet*>, Febrero 2011 [Disponible Internet, 27 de Abril 2011]

[18] Jesús Arias Fisteus, UC3M, "Servlets", <*http://www.it.uc3m.es/labttlat/material/servlets.pdf*>, 2009 [Disponible Internet, 27 de Abril 2011]

[19] Jesús Arias Fisteus, UC3M, "Java Server Pages (JSP)", <*http://www.it.uc3m.es/labttlat/material/jsp.pdf*>, 2009 [Disponible Internet, 27 de Abril 2011]

[20] Jesús Arias Fisteus, UC3M, "Desarrollo de aplicaciones Web con Servlets y JSP", <*http://www.it.uc3m.es/labttlat/material/integracion.pdf*>, 2009 [Disponible Internet, 27 de Abril 2011]

[21] Wikipedia, "Apache Tomcat", <*http://es.wikipedia.org/wiki/Apache_Tomcat*> Abril 2011 [Disponible Internet, 27 de Abril 2011]

[22] Wikipedia, "Apache Derby" <*http://es.wikipedia.org/wiki/Apache_Derby*>, Abril 2011 [Disponible Internet, 27 de Abril 2011]

[23] Nokia, "Nokia 6131: Technical specifications" <*http://europe.nokia.com/find-products/devices/nokia-6131-nfc/technical-specifications*>, 2011 [Disponible Internet, 27 de Abril 2011]

[24] Oracle, "JSRs: Java Specification Requests, JSR 257: Contactless Communication API", <http://jcp.org/en/jsr/detail?id=257>, 2011 [Disponible Internet, 27 de Abril 2011]

[25] Nokia, "Nokia 6131 NFC SDK Programmer's Guide v1.1", <*http://www.forum.nokia.com/dp?uri= http%3A%2F%2Fsw.nokia.com%2Fid%2Fc0e905fa-0436-476c-b209-96ccd44e4e9c%2FNokia_6131_NFC_SDK_Programmers_Guide_v1_1_en.pdf*>, Julio 2007 [Disponible Internet, 27 de Abril 2011]

[26] Oracle, "JSRs: Java Specification Requests, JSR-82: Java APIs for Bluetooth" <*http://jcp.org/en/jsr/detail?id=82*>, 2011 [Disponible Internet, 27 de Abril 2011]

[27] BlueCove Team, "Bluecove", <*http://bluecove.org/*>, Diciembre 2008 [Disponible Internet, 27 de Abril 2011]

Anexo A: Guía de instalación del Software

En este anexo se detalla qué software ha sido necesario para el desarrollo de este proyecto, así como detalles de su instalación y configuración.

- **Java SE Development Kit 6u16**

Es necesaria la instalación del JDK (Java Development Kit). En él reside el JRE, e incluye herramientas como el compilador de Java, Javadoc para generar documentación o el depurador.

Se puede descargar el archivo jdk-6u16-windows-i586.exe a partir del siguiente enlace: *http:///java.sun.com/products/archive/j2se/6u16/index.html*

La propia instalación, además de actualizarse automáticamente, es sencilla, puesto que sólo hay que seguir los pasos, e indicar la ruta de instalación.

- **NetBeans IDE 6.9**

La plataforma de desarrollo utilizada para realizar este proyecto es NetBeans IDE 6.9. Este entorno de trabajo proporciona todas las herramientas con las que poder trabajar.

Se puede descargar la version “All” a partir del siguiente enlace:
http://netbeans.org/downloads/6.9/

Al descargarlo, se obtendrá un archive exe, que al ejecutarlo, se mostrarán una serie de pasos para la configuración de la instalación.

A través de NetBeans, se desarrollarán las aplicaciones para Java SE, Java ME, y Java Web.

Además, se utilizarán las siguientes herramientas ofrecidas por el IDE para poder desplegar un servidor web y una base de datos.

- Apache Tomcat 6.0.26
- JavaDB Derby 10.4.2.1

NetBeans IDE 6.9 funciona sobre una Java SE Development Kit (JDK), por lo tanto, su instalación previa es requerida. Concretamente, se precisa la versión JDK 6 Update 13 o posterior.

- **Apache Tomcat 6.0.26**

El servidor Apache Tomcat se instala a partir de la instalación de Eclipse IDE. Para su uso, no es necesaria ninguna configuración. La propia aplicación Web RedBancaria, al ser ejecutada, se encarga de arrancar el servidor.

- **JavaDB Derby 10.4.2.1**

Para el caso de JavaDB Derby, será necesario agregar las siguientes librerías a las aplicaciones que la requieren, para su correcto funcionamiento: “derby.jar” y “derbyclient.jar”.

Al estar integrada con NetBeans, la configuración inicial se realiza dentro del entorno de desarrollo. Los pasos son los siguientes:
Menú de Prestaciones – Bases de datos – JavaDB – Iniciar servidor. Se indicará dónde se quieren guardar o de dónde obtener los archivos de la base de datos. Por defecto, la ruta en Windows es: “*C:\Users\Administrador\.netbeans-derby”.*
Obtendremos el siguiente mensaje:

“Se ha instalado el administrador de seguridad utilizando la directiva de seguridad de servidores básica.
Servidor de red Apache Derby - 10.4.2.1 - (706043) se ha iniciado y está listo para aceptar conexiones en el puerto 1527 a las 2011-05-01 17:07:51.788 GMT”

A continuación, en Base de datos, se elige la opción “Nueva conexión con la base de datos…”. Se selecciona en “Cambios Basicos” – Entrada URL por defecto, rellenando los siguientes campos:

- Nombre de usuario: user
- Contraseña: pass
- URL de JDBC: jdbc:derby://localhost:1527/redBancaria
- Seleccionamos el esquema APP.

En los siguientes usos, se debe sólo elegir la opción “Conectar” en esa nueva conexión directa que se acaba de crear.

- **Nokia 6131 NFC SDK 1.1**

Este SDK propietario de Nokia, permite programar aplicaciones móviles válidas para los teléfonos móviles que se usan en este proyecto. Además, es posible su integración con NetBeans, para tener así, una única plataforma de desarrollo. En el apartado 2.7.2 de este documento, se describen las características de este SDK.

Se puede descargar en el siguiente enlace, en el que también se facilita su documentación:
http://www.forum.nokia.com/info/sw.nokia.com/id/ef4e1bc9-d220-400c-a41d-b3d56349e984/Nokia_6131_NFC_SDK.html

Puede ser integrado en el NetBeans de la siguiente forma:

- En NetBeans, se hace *click* en “Herramientas – Plataformas Java”.
- Allí, se selecciona “Añadir Plataforma”.
- Se elige “*Custom Java ME MIDP Platform Emulator*”, se da a siguiente.
- En el siguiente paso, se selecciona en nuestro sistema de archivos, la carpeta donde se ha instalado el SDK Nokia, por ejemplo:

C:\Nokia\Devices\Nokia_6131_NFC_SDK_1_1, y así se detectará e instalará la plataforma de desarrollo Nokia en Eclipse, con las siguientes características:

  - *Devices* (Dispositivos):
    Nokia_6131_NFC_SDK_1_1
  - *Profiles* (Perfiles)**:**
    MIDP-2.0
  - *Configurations* (Configuraciones)**:**
    CLDC-1.1
  - *Optional APIs (APIs adicionales)***:** DBAPI-1.0, JSR184-1.0, JSR75-1.0, JSR82-1.0, MMAPI-1.0, NOKIAUI-1.0, OBEX-1.0, WMA-2.0, lib/ext/NCFIntegrationLibrary_Client.jar, lib/ext/jmf.jar, lib/ext/m2g.zip, lib/ext/nfc.zip, lib/ext/satsa.zip, lib/ext/ssl.jar, lib/ext/wsa.zip

Es importante señalar que en la configuración de la aplicación NFC Cliente, se deberá elegir: Nokia_6131_NFC_SDK_1_1, MIDP-2.0, CLDC-1.1, y el API

opcional *lib/ext/nfc.zip* (la extensión al JSR257 que nos permite la comunicación peer-to-peer.)

La configuración de la aplicación NFC Reader será: Nokia_6131_NFC_SDK_1_1, MIDP-2.0, CLDC-1.1, y los APIs opcionales *lib/ext/nfc.zip* y JSR82-1.0 (que nos permite la comunicación Bluetooth).

- **Librería Bluecove 2.1.0**

Esta librería se detalla en el apartado 3.2.2. Se trata de una librería de Java para Bluetooth basada en la implementación JSR-82, para una plataforma Java SE.

Esta librería se ha de añadir en el proyecto de la aplicación del servidor PC, para posibilitar su comunicación Bluetooth con otro dispositivo. En este caso se ha utilizado la versión Bluecove 2.1.0.

La librería Bluecove se puede descargar de la siguiente página, donde también se encuentra la documentación de la misma:
http://bluecove.org/